

Chapman University

Chapman University Digital Commons

Computational and Data Sciences (Ph.D.)
Dissertations

Dissertations and Theses

Winter 12-4-2019

Image Restoration using Automatic Damaged Regions Detection and Machine Learning-Based Inpainting Technique

Chloe Martin-King

Chapman University, marti192@mail.chapman.edu

Follow this and additional works at: https://digitalcommons.chapman.edu/cads_dissertations



Part of the [Artificial Intelligence and Robotics Commons](#), [Numerical Analysis and Scientific Computing Commons](#), [Other Analytical, Diagnostic and Therapeutic Techniques and Equipment Commons](#), [Other Applied Mathematics Commons](#), [Partial Differential Equations Commons](#), [Pathology Commons](#), and the [Signal Processing Commons](#)

Recommended Citation

C. Martin-King, "Image restoration using automatic damaged regions detection and machine learning-based inpainting technique," Ph.D. dissertation, Chapman University, Orange, CA, 2019. <https://doi.org/10.36837/chapman.000114>

This Dissertation is brought to you for free and open access by the Dissertations and Theses at Chapman University Digital Commons. It has been accepted for inclusion in Computational and Data Sciences (Ph.D.) Dissertations by an authorized administrator of Chapman University Digital Commons. For more information, please contact laughtin@chapman.edu.

Image Restoration using Automatic Damaged Regions Detection and Machine Learning-Based Inpainting Technique

A Dissertation by

Chloe Furness Martin-King

Chapman University

Orange, CA

Schmid College of Science and Technology

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computational and Data Sciences

December 2019

Committee in charge:

Mohamed Allali, Ph.D., Chair

Erik Linstead, Ph.D.

Hesham El-Askary, Ph.D.

Mohammad Kamal, M.D.



CHAPMAN UNIVERSITY
SCHMID COLLEGE OF SCIENCE AND TECHNOLOGY

Computational and Data Sciences

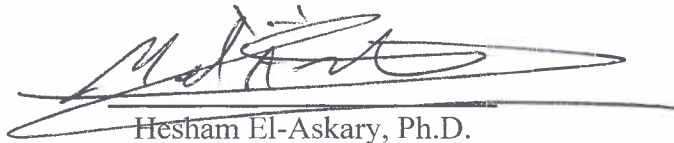
The dissertation of Chloe Furness Martin-King is approved.

A handwritten signature in black ink, appearing to read 'Mohamed Allali', written over a horizontal line.

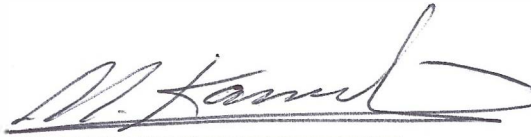
Mohamed Allali, Ph.D., Chair

A handwritten signature in black ink, appearing to read 'Erik Linstead', written over a horizontal line.

Erik Linstead, Ph.D.

A handwritten signature in black ink, appearing to read 'Hesham El-Askary', written over a horizontal line.

Hesham El-Askary, Ph.D.

A handwritten signature in black ink, appearing to read 'Mohammad Kamal', written over a horizontal line.

Mohammad Kamal, M.D.

December 2019

Image Restoration using Automatic Damaged Regions Detection and Machine Learning-Based Inpainting Technique

Copyright © 2019

by Chloe Furness Martin-King

ACKNOWLEDGEMENTS

Thank you to my mom for genuinely thinking that I am smart and believing in me though you did not know exactly what I was doing. You have been my biggest fan my entire life. I love you to the moon and back. Thank you to my daughter, Emma, for being my buddy and a source of light. Thank you to my husband, Michael, for all the food.

Thank you to my advisor, Dr. Allali, for introducing me to the beautiful field of digital image processing. You knew what I was doing, and you still believed in me. I could not have asked for a better advisor. You gave me space to have my own ideas and did not discourage me. You were the voice of reason during some very heavy and trying times. I am not sure why you do not think I am nuts.

Thank you to Dr. Linstead for introducing me to machine learning, particularly convolutional neural networks, and for assigning some of the most enjoyable homework.

Thank you to Dr. El-Askary for the many opportunities you have given to me as the program director of CADs and as a member of my committee.

Thank you to Dr. Kamal and OmniPathology for providing quality slide images and taking the time to share your expertise. Also, I think it is cool that you are both a pathologist and a visual artist.

Thank you to James Kelly for your patience and quick responses to my e-mails about the cluster. Thank you to Dr. Fahy for making an account for me on the cluster and advising me to take a programming course before beginning the CADs program. I chose C++ and it helped immensely.

Thank you to Dana Dacier and Robin Pendergraft for your tireless and genuine assistance.

Thank you to Bill and Joan for allowing me to take up an entire room at your house to work. You were so kind and gracious. Sorry I brushed my teeth in the kitchen sink four times.

Thank you to my dad, Dr. Martin, for believing that I can do anything. I miss you more than you could possibly know, and I love you.

Thank you to my dog and shadow, Yakimo, for making everything easier.

Most of all, thank you, God, my favorite mathematician. Thank you for holding me together and being my constant uplifting companion.

TECHNICAL ACKNOWLEDGMENTS

Chapters 2 and 3 include content, written by Chloe Martin-King, from our previously published conference paper, *Automatic Damaged Region Detection and Inpainting Method for Digital Images*, which was presented July 27, 2016 in Las Vegas, NV [1]. Copyright © 2016 CSREA Press.

Chapters 4 and 5 include content, written by Chloe Martin-King, from our conference paper, *Region Hiding for Image Inpainting via Single-Image Training of U-Net*, which was presented December 7, 2019 at CSCI 2019 in Las Vegas, NV. The conference proceedings will be published on IEEE Xplore in 2020 [2]. Copyright © 2019 IEEE.

DEDICATION

To my mom. Thank you and I love you.

ABSTRACT

Image Restoration using Automatic Damaged Regions Detection and Machine Learning-Based

Inpainting Technique

by Chloe Furness Martin-King

In this dissertation we propose two novel image restoration schemes. The first pertains to automatic detection of damaged regions in old photographs and digital images of cracked paintings. In cases when inpainting mask generation cannot be completely automatic, our detection algorithm facilitates precise mask creation, particularly useful for images containing damage that is tedious to annotate or difficult to geometrically define. The main contribution of this dissertation is the development and utilization of a new inpainting technique, region hiding, to repair a single image by training a convolutional neural network on various transformations of that image. Region hiding is also effective in object removal tasks. Lastly, we present a segmentation system for distinguishing glands, stroma, and cells in slide images, in addition to current results, as one component of an ongoing project to aid in colon cancer prognostication.

TABLE OF CONTENTS

	<u>Page</u>
ACKNOWLEDGEMENTS	IV
TECHNICAL ACKNOWLEDGMENTS.....	VI
DEDICATION.....	VII
ABSTRACT.....	VIII
LIST OF TABLES	XI
LIST OF FIGURES	XII
LIST OF ABBREVIATIONS	XVII
LIST OF SYMBOLS	XVIII
1 INTRODUCTION.....	1
2 TRADITIONAL AND NON-MACHINE LEARNING-BASED IMAGE RESTORATION	3
2.1 Basic Inpainting Using Heat Transfer Equations.....	3
2.2 Texture Synthesis and Patch Based Image Interpolation.....	6
2.3 Total Variation in Image Denoising and Inpainting	9
2.4 Homogeneous Diffusion Inpainting for Image Compression.....	10
3 AUTOMATIC DAMAGED REGIONS DETECTION ALGORITHM AND RESTORATION OF DAMAGED PHOTOGRAPHS	15
3.1 Automatic Crack Detection in Photographed Scenes using Singular Value Decomposition	15
3.2 Our Improved Detection Method.....	17
3.3 Experimental Results	18
3.4 Utilizing the Inverted Laplacian Magnitude for Image Compression in Damaged Regions Inpainting Tasks.....	30
3.5 Conclusion	35
4 MACHINE LEARNING-BASED IMAGE RESTORATION	36
4.1 Loss	37
5 REGION HIDING FOR IMAGE INPAINTING VIA SINGLE-IMAGE TRAINING OF U-NET.....	40
5.1 Introduction.....	40

5.2	Approach.....	41
5.2.1	Region Hiding.....	41
5.2.2	Set-Up	42
5.2.3	Data Augmentation	45
5.2.4	Loss	46
5.2.5	Architecture.....	47
5.3	Implementation	48
5.4	Software and Hardware.....	51
5.5	Results.....	52
5.5.1	Quantitative.....	55
5.5.2	Qualitative.....	56
5.5.3	Ablation Study	58
5.6	Conclusion	58
6	SEGMENTATION OF STRUCTURES IN WHOLE SLIDE IMAGES OF COLON TISSUE WITH U-NET.....	62
6.1	Introduction.....	62
6.2	Background and Existing Knowledge	63
6.3	Segmentation Project Aims.....	66
6.4	Data	67
6.5	Approach.....	68
6.6	Gland Segmentation U-Net Architecture.....	71
6.7	Results.....	73
6.7.1	Quantitative Assessment of U-Net Gland Segmentation Results	73
6.7.2	Qualitative Assessment of Cell Isolation Results	79
6.8	Software and Hardware.....	85
6.9	Cell Type Classification.....	86
7	CONCLUSION	89
	REFERENCES.....	92
	APPENDICES.....	100

LIST OF TABLES

	<u>Page</u>
Table 5-1: Partial Convolutional Network Architecture.....	48
Table 5-2: Quantitative Inpainting Results	55
Table 6-1 Details of Gland Segmentation U-Net Architecture.....	72
Table 6-2 Metric values for test sets A and B of GlaS Warwick-QU.....	75
Table 6-3 Quantitative comparison of three segmentation mask overlap samples.....	76
Table 6-4 Images of cell types	86
Table B-1 Letter names associated with pixel locations.....	111
Table B-2 Partial differential equations in terms of individual pixels.....	112

LIST OF FIGURES

	<u>Page</u>
Figure 2-1 Visual representation of inpainting applied to a rectangular region [1].....	5
Figure 2-2 Basic texture synthesis algorithm diagram. Patches ua , ub , and uc correspond to candidate pixels, a , b , and c , respectively. The missing or damaged target pixel, i , is assigned the value of pixel c based on the similarity metric determining uc to be the best match for u .7	
Figure 2-3 Texture synthesis with PDE-based inpainting results on Barbara image [8]. From top to bottom, left to right: Damaged input image, inpainted structure image, final restoration result from adding the structure and texture images together, and texture synthesized texture image.	8
Figure 2-4 From left to right, (a) Input image to be compressed [15], (b) results from applying smoothed Laplacian magnitude inverse, and (c) results from applying Floyd-Steinberg error diffusion dithering to the inverted Laplacian magnitude in (b).	11
Figure 2-5 Inverted Laplacian magnitude after thresholding.	14
Figure 3-1 From left to right, top to bottom: (a) Image of cracked pavement [19], (b) results from applying our automatic detection algorithm, (c) structure image after inpainting (b) with simple diffusion, (d) final result from adding the inpainted structure (c) and synthesized texture (e) images, (e) texture image after applying texture synthesis with neighborhood size of 15, (f) texture image before applying texture synthesis, (g) enlarged final result with selected region outlined in green and close-ups of the input image, the synthesized texture image, the ..	20
Figure 3-2 From left to right, top to bottom: (a) Image of cracked pavement [20], (b) results from applying our automatic detection algorithm, (c) structure image after inpainting (b) with simple diffusion, (d) final result from adding the inpainted structure (c) and synthesized texture (e)	21
Figure 3-3 From left to right, top to bottom: (a) Close-up of an old painting with very fine cracks [21], (b) automatic detection results using our SVD algorithm with $\delta = 0.9911mn$, (c) restoration results from applying inpainting, (d) restoration results from applying TV denoising to the entire image, (e) restoration results from applying isotropic Gaussian smoothing kernel with standard deviation of 2, and (f) same as (a) with two selected regions outlined in green to indicate the locations of the areas examined in Figure 3-4.....	25
Figure 3-4 From left to right: Two regions of interest as established in part (f) of Figure 3-3 and their corresponding detection and restoration results, TV denoising results, and Gaussian smoothing results.	26
Figure 3-5 Two 9×9 blocks taken from the same location in different restoration versions of the painting in Figure 3-4. The block on the left is taken from the inpainting results without applying boundary condition equations at the borders of the image. The block on the right is taken from	

the inpainting results achieved with utilizing boundary condition equations at the borders of the image. Notice that the pixels at indices 2,9 and 9,6 of the left block correspond to cracks that are resolved in the block on the right. The block is located at the bottom right of the input image. Graphic created in MATLAB with *imagesc* function [18]...... 27

Figure 3-6 Three sets of images corresponding to three damaged photographs [22] [23] [24]. From left to right: Input damaged photograph, restoration results from using the partially user dependent method of mask creation method, automatic damaged regions detection results, and modified mask which is used to inpaint the damaged photograph in the first column..... 29

Figure 3-7 Image with damage that is characterized by light colored spots throughout and concentrated over the face of the subject [25]. 30

Figure 3-8 Input image with selected region inscribed in a red square and the corresponding close-up revealing that the damage is characterized by dots and splotching..... 31

Figure 3-9 SVD-based automatic detection and inpainting results. Notice the blurred edge of the girl's right cheek, chin, and eyes in part (e)..... 31

Figure 3-10 Inverse Laplacian magnitude automatic detection mask results. The edges of the girl's right cheek, chin, and eyes are preserved. 32

Figure 3-11 Demonstration of how the Laplacian magnitude mask creates lines that straddle the edges rather than directly on top of the edges. Sobel edge detection in the image on the right indicates the location of the edge that the mask in the image on the left outlines..... 33

Figure 3-12 Raw results from using the SVD detection and inverted Laplacian magnitude detection (without mask modifications). The two top images demonstrate the difference between the two detection methods. The top image on the left shows blurring in areas where edges exist and should be preserved whereas the top right image shows edges remain uncompromised. The two bottom images are the inpainting masks detected using SVD and Laplacian magnitude, respectively. Zoom in for better viewing..... 34

Figure 5-1 From left to right, ground truth image (I_0), damaged image (I) with damage shown as white, and binary mask representing locations of artificially damaged pixels in black. 42

Figure 5-2 Two instances of transformation and mask generation. From left to right, transformed damaged image, mask such that damage is not included, and transformed damaged image with mask applied (IM). 43

Figure 5-3, Inpainting results for five images. From left to right, (a) damaged images, (b) inpainting results from applying PatchMatch [10], (c) inpainting results from applying multi-image trained PConvNN [34], (d) inpainting results from training PConvNN on the single damaged image using our region hiding technique, partial convolution based padding [48], and local scaling, (e) inpainting results from training PConvNN on the single damaged image using our region hiding technique, zero padding, and global scaling, and (f) ground truth image. 54

Figure 5-4 Three sets of damaged images with selected regions outlined in green. Close-up images from left to right: selected damaged regions, PM results, PConvNN trained on ImNet results, PConvNN trained on single image using PConv padding and local scaling, PConvNN trained on single image using zero padding and global scaling, and ground truth. Zoom in for better viewing..... 57

Figure 5-5 Columns from left to right: Close-ups of selected damaged regions, results from training without perceptual loss, results from training without style loss, results from training with all loss terms, corresponding ground truth regions. Zoom in for better viewing. 58

Figure 5-6 From left to right, top to bottom, original image [56], normalized reference image, original image with unwanted object masked, object removal result from training on the reference image to inpaint the masked image..... 60

Figure 6-1 Example of WSI of colon tissue sample [66]. Top left: Macro image of the entire slide with three sections of interest indicated with a green rectangle. Bottom left: Thumbnail of three sections from the same tissue sample with a red box around a region of interest within the left-most section. Right: close-up of the region of interest from the actual WSI..... 64

Figure 6-2 WSI of healthy colon tissue from the OmniPathology dataset [67]. Image on left is at 10X magnification and image on right is at 20X magnification. Black rectangle in left image indicates location of the image on the right. Stroma between the glands appears normal with no inflammatory cells. The glands are one-cell thick and appear organized..... 65

Figure 6-3 Comparison of structures in healthy and unhealthy colon tissues [73]. Left: healthy glands with normal stroma and organized epithelial cells. Right: Malignant glands with stroma containing inflammatory cells and disorganized masses of epithelial cells which have become tumors. When stroma is created in response to cancer, it is called the stromal response, or desmoplasia. 66

Figure 6-4 An example of a histogram for a region of interest from a WSI. The first threshold is based on the mean pixel value and the second threshold is based on the first zero crossing that occurs after the global maximum. This can be more easily appreciated in the figure on the right which incorporates a plot of the derivative of the histogram below. The slope of the histogram equals zero at pixel value 231..... 68

Figure 6-5 The histograms of three normalized images from the OmniPathology dataset [67]. The vertical red bars indicate the pixel values that delineate the three main components of the input images: (1) Glands and cells in the stroma, (2) stroma, and (3) background artifact. The first threshold value is equivalent for each image due to the color normalization operation. The second threshold value corresponds to the local minimum immediately following the absolute maximum of the histogram. 70

Figure 6-6 Three samples to show the difference between predicted segmentation results and ground truth segmentation. White indicates overlap of the two segmentation masks (true positives), purple indicates regions that the ground truth mask considers glandular tissue but that the prediction mask does not (false negatives), and green indicates regions that the prediction mask considers glandular tissue but that the ground truth mask does not (false positives). We

have also included corresponding pixel level metrics and object level metrics in Table 6-3.	76
Figure 6-7 Successful segmentation results for five images taken from test set A of [73]. The left column shows the input images, the center column shows segmentation prediction mask atop the input image, and the right column shows the ground truth segmentation mask atop the input image.	78
Figure 6-8 Segmentation results from using our histogram thresholding algorithm on the three images in Figure 6-5. The top two images are at 20X magnification while the bottom image is at 10X magnification. All three images are from the OmniPathology dataset [67].	80
Figure 6-9 Histogram of normalized image of benign tissue from test set B of [73] and segmentation results from utilizing our automatic thresholding algorithm. The vertical red bars indicate the pixel values that delineate the three main components: (a) Glands and cells in stroma, (b) stroma, and (c) background. Since there is no local minimum after the global maximum, which occurs at pixel value 255, a default value of 216 is assigned to the second threshold.	81
Figure 6-10 Example of utilizing the two segmentation methods together to isolate various structures in a region of interest taken from a WSI from the CDSA [66]. From here, we can distinguish between cells in the stroma and cells in the tumors.	82
Figure 6-11 Cell nuclei isolation. Top left to bottom right: Input image, results from filtering relatively <i>light</i> pixels from the CIE L*a*b* transformed input image, application of the mask in part (f) of Figure 6-10 to show isolated cell nuclei located in the tumors, and application of the mask in part (g) to show isolated cell nuclei located in the stroma.	83
Figure 6-12 Close-up of cells in the stroma. The rightmost image is the result from isolating cell nuclei in the center image.	84
Figure 6-13 Close-up of cells in the stroma taken from the segmentation results shown in Appendix C. The rightmost image is the result from isolating cell nuclei in the center image. Depending on the application, the center image may be more desirable than the image on the right.	84
Figure 6-14 Regions of interest are subjected to color normalization, histogram analysis, and preliminary segmentation before being separated into sub-images and sent to pathologists for annotation.	87
Figure 6-15 From left to right: Sub-image from region of interest outlined in red in Figure 6-14, automatic thresholding results, and <i>simulated</i> semantic segmentation of tissue by cell type. (Important: The image on the right is not an actual semantic segmentation result.)	88
Figure 6-16 Simulated cell counting results.	88
Figure B-1 Pixel names and locations.	111

Figure B-2 Sample pixel values and locations corresponding to diagonal edge.	113
Figure C-1 Segmentation of cell nuclei using our automatic method. From left to right, top to bottom: (a) Slide background and very thin biopsy tissue, (b) stromal tissue with individual cells masked, (c) glands and cells in stroma, (d) isolated cell nuclei within the glands and stroma, and (e) further isolated cell nuclei solely within the glands.	119
Figure C-2 Two-part segmentation of cell nuclei using our automatic method. From left to right, top to bottom: (a) Region of interest taken from OmniPathology dataset [67] at 20X magnitude, (b) gland segmentation results from trained U-Net, (c) non-glandular structures, (d) stromal tissue with individual cells masked, (e) glands and cells in stroma, (f) slide background, (g) cell nuclei within the glands and stroma, (h) further isolated cell nuclei solely within the glands/tumors, and (i) cell nuclei solely within the stroma. See Figure 6-13 for a close-up comparison of (e) and (i).....	120
Figure C-3 From left to right, top to bottom: (a) Region of interest taken from OmniPathology dataset [66] at 20X magnitude, (b) gland segmentation results from trained U-Net, (c) non-glandular structures, (d) stromal tissue with individual cells masked, (e) glands and cells in stroma, (f) cells in stroma (tumors masked).	121
Figure C-4 From left to right: Close-up of input image in Figure C-3, cell segmentation using histogram thresholding algorithm, and cell nuclei segmentation by filtering out light pixels in CIE $L^*a^*b^*$ color space. Both segmentation images are useful depending on the task. Smooth muscle cells are visible in the center image and masked in the second, which favors cells with dark nuclei.	121
Figure C-5 Histogram and segmentation results for normalized image of benign colon tissue. Image taken from test set A of [73].	122
Figure D-1 U-Net-like architecture schematic for region hiding system.	123

LIST OF ABBREVIATIONS

<u>Abbreviation</u>	<u>Meaning</u>
CNN	Convolutional Neural Network
\mathcal{L}	Loss
PDE	Partial Differential Equation
PSNR	Peak Signal to Noise Ratio
ROF	Rudin, Osher, and Fatemi, in reference to their important paper on utilizing total variation to denoise images.
SMC	Smooth Muscle Cell
SSIM	Structural Similarity Index
TV	Total Variation

LIST OF SYMBOLS

<u>Symbol</u>	<u>Meaning</u>
$Div(\cdot)$	Divergence
∇f	Gradient of f , $\nabla f = \frac{\partial f}{\partial x} \mathbf{i} + \frac{\partial f}{\partial y} \mathbf{j} = \left\langle \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right\rangle$
G	Gram matrix
\cap	Intersection
Δf	Laplacian of f , $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
\mathcal{L}	Loss
ℓ_1	ℓ_1 loss, calculated $\sum_{i=1}^n y(i)_{true} - y(i)_{predicted} $
$\Delta^\perp f$	Orthogonal gradient of f (isophote direction), $\Delta^\perp f = \left\langle -\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x} \right\rangle$.
\odot	The Hadamard product operator. Elementwise multiplication of two matrices.
A^T	Transpose of matrix A
\cup	Union

1 Introduction

Image restoration and manipulation have been popular since the advent of photography. What was once meticulously done by hand is now possible to do digitally and automatically. Image inpainting tasks embody the many ways in which missing or unwanted regions in an image are repaired. Just as a museum's art restoration expert will use intact content to renovate damaged areas in a painting, digital image inpainting utilizes extant pixel values to calculate and assign new values to damaged or missing pixels. Many traditional inpainting methods rely on boundary conditions and partial differential equations (PDEs) to model and discretely implement diffusion such that neighboring pixel values are propagated into the interior of the damaged region. Methods that do not rely on PDEs, such as nonlocal, exemplar-based inpainting, produce markedly improved results over their locally restricted counterparts. However, both PDE and non-PDE-based inpainting models are often supplemented with preprocessing and postprocessing techniques to improve visual results. Furthermore, even clever combinations of systems fall short because these methods depend solely on a single image to provide adequate material.

To overcome this issue, computational and image processing scientists have turned to machine learning, particularly, convolutional neural networks (CNNs). The typical requirements of such systems are access to many training and validation samples, time and resources to train a CNN, and/or access to appropriate open source pretrained weights. Additionally, multiple images with similar content to the damaged image may be difficult to find, if available at all. Nonetheless, machine learning-based inpainting systems provide impressive results. Although

access to large, diverse, image repositories significantly contributes to the success of CNNs in inpainting tasks, especially if repairing images containing unique features, like faces, it is a collaboration of several factors that produces such compelling predictions. Admittedly, CNNs can train on relatively few images and perform remarkably well on segmentation and classification tasks [3] [4].

In this dissertation we detail a thorough investigation of image restoration techniques, propose a novel inpainting technique using machine learning, and expand image processing into the field of computational pathology to aid in colon cancer prognostication based on biopsy slide images. A desire to find an interesting and meaningful problem to apply our prospective solutions to led to a timely and important collaboration of image processing, deep learning, and computational pathology. This dissertation represents a melding of solution, problem, and real-world application. The journey began as an interest in image processing, particularly, image restoration using inpainting and texture synthesis.

We will first discuss previous and current research pertaining to image restoration before transitioning into our automatic damaged regions detection algorithm. Then we will describe our novel inpainting technique: region hiding. Next, we will introduce computational pathology, important biological aspects and background information pertaining to colon cancer, explain the problem that our work aims to address, and describe the segmentation processes. Lastly, we have included derivations of meaningful PDEs for use in discrete image restoration algorithms in Appendices A and B, additional biopsy slide image structures segmentation results in Appendix C, and the CNN architecture used with region hiding for inpainting in Appendix D.

2 Traditional and Non-Machine Learning-Based Image Restoration

2.1 Basic Inpainting Using Heat Transfer Equations

One classic PDE-based inpainting method relies on the mathematical description of heat transfer through a solid: $u_t = u_{xx} + u_{yy}$, where $u(x, y, t)$ is the temperature at location (x, y) and time t . In image processing applications, the heat transfer equation can be represented discretely using second order Taylor series approximations. Missing pixel values are calculated using the values of neighboring intact pixels. Once a missing pixel's value is assigned, it can be used to calculate other missing pixel values. An iterative approach, which is addressed by the time variable t in the heat transfer equation, allows a pixel that has already been assigned a value in a previous iteration to be assigned a new value. This is based on a series of *updater* steps in which the last value of a target pixel is used in tandem with the nonchanging value of a neighboring pixel to calculate the new value of the target pixel.

Inpainting a digital image requires that the heat transfer equations be treated discretely as difference equations. Before difference equations are applied to the target region, a two-pixel thick boundary immediately surrounding the region is utilized to interpolate the initial one-pixel thick interior of the region. In image processing tasks, two-dimensional image pixel locations are represented in the familiar coordinate format, (x, y) . However, pixel indices signify locations that are different from the x and y values on a standard two-dimensional coordinate plain. Starting from the top, left corner of the image, $(0,0)$, the first value, x , indicates how many

pixels downward from the origin the location is. The second value, y , indicates how many pixels to the right of the origin the location is. To force the information from the outside of the region into the inside of the region, given edge location in relation to the origin of the image, the discrete double derivative equations must be altered counter intuitively. As shown in Figure 2-1, for a two dimensional image, $I = I(x, y)$, the top and left edges may be approximated using equations (2.1) and (2.2). However, for the bottom and right edges, the application is reversed, as in equations (2.3) and (2.4).

Top edge:

$$I(x, y) = 2I(x - 1, y) - I(x - 2, y), \quad (2.1)$$

where $x = i$ is constant and $y = j:j + n$.

Left edge:

$$I(x, y) = 2I(x, y - 1) - I(x, y - 2), \quad (2.2)$$

where $y = j$ is constant and $x = i:i + m$.

Bottom edge:

$$I(x, y) = 2I(x + 1, y) - I(x + 2, y), \quad (2.3)$$

where $x = i + m$ is constant and $y = j:j + n$.

Right edge:

$$I(x, y) = 2I(x, y + 1) - I(x, y + 2), \quad (2.4)$$

where $y = j + n$ is constant and $x = i : i + m$. We have included the derivations of these discrete boundary equations using boundary conditions in Appendix A.

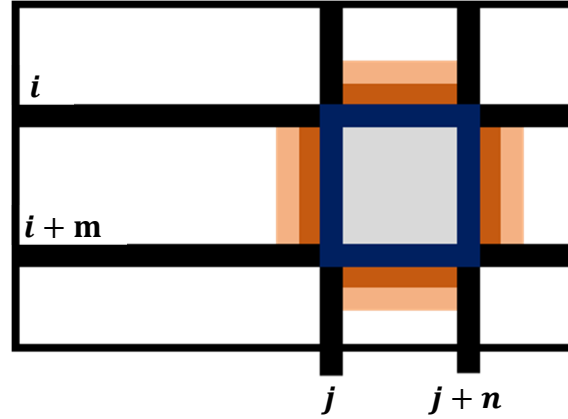


Figure 2-1 Visual representation of inpainting applied to a rectangular region [1].

To inpaint the interior of the damaged or missing region, we apply diffusion within the boundary using the discrete representation of the heat transfer equation, $f_t = f_{xx} + f_{yy}$.

Introducing the third variable, t , which is the time component relevant to the heat-transfer process, allows us to iteratively reassign values within the interior using diffusion-based image inpainting. Although the time variable is in the third position of the function's input tuple, which is often associated with channel indices, we limit our illustration to two dimensional images and therefore recognize t as descriptive of varying time steps rather than switching through color channels.

For a two-dimensional image, $I = I(x, y, t)$, solving for $I(x, y, t + r)$ in the discrete representation of the heat transfer equation gives us

$$I(x, y, t + r) = I(x, y, t) + r(I(x + 1, y, t) + I(x - 1, y, t) + I(x, y + 1, t) + I(x, y - 1, t) - 4I(x, y, t)), \quad (2.5)$$

where the step size r is a small number between 0 and 1 and $i + 1 \leq x \leq i + m - 1$ and $j + 1 \leq y \leq j + n - 1$. In practice, a large number of steps in tandem with the small step size is used to gradually fill in the missing information in an iterative process descriptive of heat transfer over time. We have also included the derivation of the discrete heat transfer equation in Appendix A.

2.2 Texture Synthesis and Patch Based Image Interpolation

Simple PDE-based inpainting models are easy to implement and most effective with small inpainting regions. When inpainting regions encompass patterned objects, diffusion can be paired with texture synthesis to improve inpainting results [5] [6]. The authors of [5] propose an impressive scheme in which the target image is separated into two components; texture and structure. The texture component is inpainted using a straight-forward texture synthesis procedure such that missing or damaged pixels are assigned values based on a set of accessible nearby pixels. The similarity metric used in [5], per the suggestion of [7], is the normalized sum of squared differences, $\sum(u - \hat{u})^2$, where u is an $m \times n$ patch of pixels adjacent to the missing or damaged pixel of interest, and \hat{u} is an $m \times n$ patch of pixels within a specific neighborhood distance of the missing or damaged region. The algorithm is illustrated by the diagram in Figure 2-2 as similarly presented in [5].

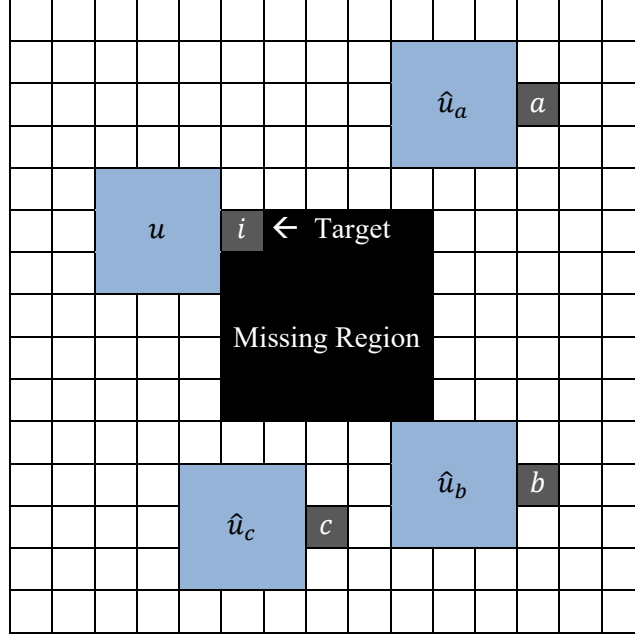


Figure 2-2 Basic texture synthesis algorithm diagram. Patches \hat{u}_a , \hat{u}_b , and \hat{u}_c correspond to candidate pixels, a , b , and c , respectively. The missing or damaged target pixel, i , is assigned the value of pixel c based on the similarity metric determining \hat{u}_c to be the best match for u .

The authors utilize high-order PDEs for image inpainting to fill in damaged or missing regions of the structure component of the original image. This is a common method of propagating known boundary information (neighboring pixel values) into missing regions so that isophote directions are heeded and edge locations and their corresponding contrasts are preserved. Isophote lines delineate pixels of a certain intensity from pixels of another intensity in adjacent regions. Visually, this translates to compartmentalizing areas that are subjected to the same amount of light. Regarding isophote directions can be just as important as edge preservation in image restoration. Numerically solving $\frac{\partial I}{\partial t} = \nabla(\Delta I) \cdot \nabla^\perp I$, where $\nabla(\Delta I)$ is the Laplacian of the gradient of I and $\nabla^\perp I$ is the orthogonal gradient of I , effectively achieves this propagation.

To overcome the texture synthesis algorithm weighing each distance equally in its discernment of similarity regardless of the location of the prospective pixel with respect to the

damaged pixel, [7] proposes applying a Gaussian kernel to the sum of squared distances value. Figure 2-3 shows the two image components after applying the associated texture synthesis and structure inpainting. The image on the bottom left is the result from adding the two repaired components. Both structure and texture are represented appropriately.

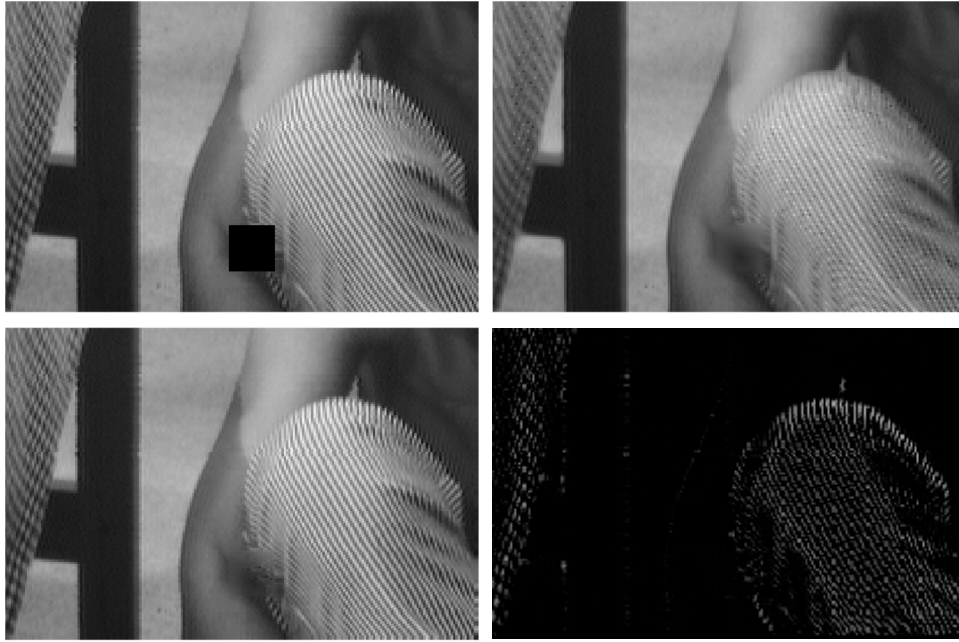


Figure 2-3 Texture synthesis with PDE-based inpainting results on Barbara image [8]. From top to bottom, left to right: Damaged input image, inpainted structure image, final restoration result from adding the structure and texture images together, and texture synthesized texture image.

Other methods, such as [9], aim to overcome the familiar challenge of interpolating underlying textures and patterns present in missing regions by employing high dimensional model representations with Lagrange interpolation to generate candidate inpainted regions, then selecting the best inpainted region to fill the missing region. The algorithm selects the best pixel intensity for the damaged region by considering pixel intensity changes surrounding the damaged region and exploiting patch rotation to determine the most appropriate candidates. PatchMatch [10], although not specifically developed for the purpose of inpainting, is regarded as one of the leading non-machine learning methods of real-time digital image restoration. The algorithm

generates a matrix of distance values in which each element represents the distance between a patch in image A and its nearest neighbor patch in image B . In the context of inpainting, possible solutions for the missing regions of image A are determined by iteratively searching for similar patches in the nondamaged regions of A .

2.3 Total Variation in Image Denoising and Inpainting

Total variation (TV) regularization is featured in many image restoration tasks because of its ability to suppress noise yet preserve edges and maintain overall structure. It was initially introduced as an essential component of the nonlinear noise removal algorithm proposed in the famous Rudin-Osher-Fatemi (ROF) paper [11]. To separate the structure and texture components in Figure 2-3, we applied TV denoising to the entire input image to obtain the structure component then subtracted this from the input image to obtain the texture component. For interested readers, we have included derivation of the discrete TV flow equation as well as exploration of how applying TV directly affects pixel values in Appendix B. Complementing diffusion or other inpainting techniques with a TV regularization term manifests as a smoothing quality within the result.

Digital image compression and transmission are processes in which data loss can occur. In [12], the authors regard digital images as vector valued functions and utilize TV minimization to restore images that have been compromised by lossy compression for transmission or communication. The inpainting process is guided by the wavelet domain rather than the pixel domain. This is motivated by the fact that the JPEG2000 image compression standard relies on the discrete wavelet transform (DWT). In the pixel domain, it is necessary to assume a decoupled relationship between pixels that are a threshold distance away from each other for the purpose of

denoising. However, wavelets inpainting requires a forced relationship between wavelet regularities to correlate the missing and existing components.

Damage within the wavelet domain propagates throughout the spatial domain with varying degrees of degradation. This implies that the damage is spatially inhomogeneous. It would be unreasonable to inpaint within the spatial domain because the corruption cannot be geometrically defined and repaired. The boundaries of the damaged regions are indistinct and diffusion-based inpainting would introduce noise since corrupted pixels would be used to repair missing pixels. The algorithms in [12] restore missing wavelet coefficients and repair corrupt wavelet packets, manifesting as properly restored edges and contours in the spatial domain. In a noiseless image, the first model proposed in [12] fills the missing wavelet coefficients using TV minimization. Alternatively, if the image is noisy, a second model for wavelet inpainting is used. The second model differs from the first in that it includes an additional term; the weighted sum of squared differences between the stored wavelet coefficients of the corrupted image and the predicted wavelet coefficients. The weight is equivalent to zero in the inpainting regions of the wavelet domain and equal to a positive constant otherwise. In section 2.4, we discuss how inpainting is applied to the spatial domain after compression to decode an encoded image.

2.4 Homogeneous Diffusion Inpainting for Image Compression

In this section we describe the use of PDEs-based inpainting in compressed image restoration by concentrating on the methods explored in [13] and [14]. “Edge-Based Compression of Cartoon-like Images with Homogeneous Diffusion,” employs PDEs-based inpainting for compressed image reconstruction [13]. “Optimising Spatial and Tonal Data for PDE-Based Inpainting,” authored in part by two corresponding authors of [13], also employs

homogeneous diffusion inpainting for compressed image reconstruction but with an added focus on spatial and tonal optimization [14].

Pixel values that lie directly on top of an edge in an image are not useful for decompression since edges often delineate structures with drastically different pixel values. Therefore, the compression method in [13] stores the pixel values on both sides of an edge. The compressed image is stored in two main components, an encoded edge map and a vector of encoded key pixel values. A header file to inform the decompression unit which coders were used to encode specific details of the image is employed together with the edge and pixel value data. In the last step, homogeneous diffusion is used to reconstruct regions between the edges. Much of the process is outside of the scope of this dissertation. However, for interested readers, intuitive and detailed algorithms and discussions are provided in [13].

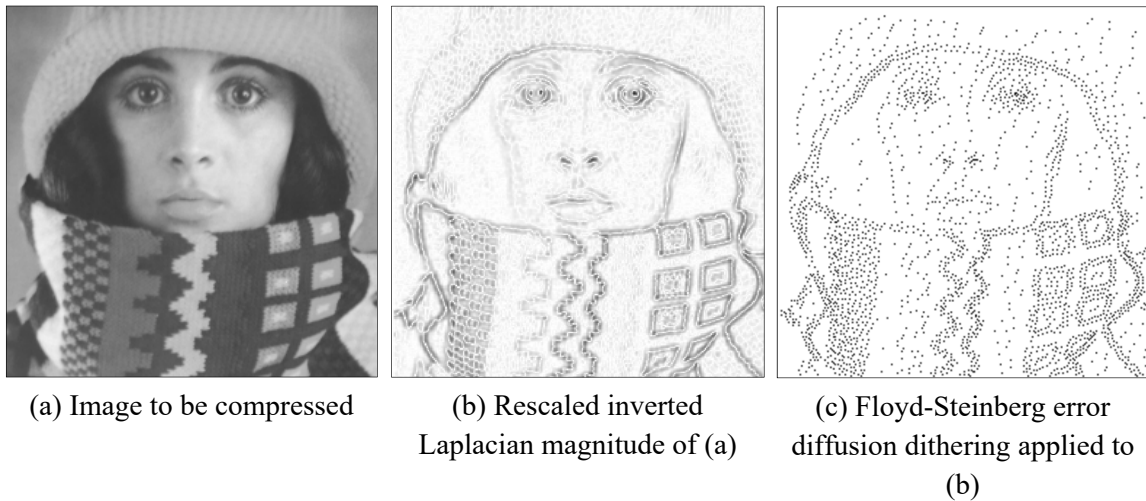


Figure 2-4 From left to right, (a) Input image to be compressed [15], (b) results from applying smoothed Laplacian magnitude inverse, and (c) results from applying Floyd-Steinberg error diffusion dithering to the inverted Laplacian magnitude in (b).

Our results from applying the smoothed Laplacian magnitude inverse to the original image, then applying Floyd-Steinberg error diffusion dithering [16] to the inverted Laplacian

magnitude as suggested in [14], are shown in Figure 2-4. Utilizing the same parameter values as [14] with Gaussian pre-smoothing standard deviation $\sigma = 1$, dithering parameter $s = 0.8$, and a mask pixel density of 4%, or $d = 0.04$, the rescaled Laplacian magnitude is:

$$\frac{.04 * \max(I)}{\text{mean}(|\Delta I_G|^{0.8})} * |\Delta I_G|^{0.8}, \quad (2.6)$$

where I_G is the Gaussian smoothed version of the image to be compressed, I , and $\max(I)$ is the maximum possible pixel value in I . In [14], the authors implement electrostatic halftoning to attain a binary point mask that preserves the average pixel value at each location. Next, the authors create an inpainting mask to preserve prime pixel locations then fine-tune the grayscale pixel values.

Because the missing region is not what is typically associated with inpainting tasks, i.e., the region to be restored exists throughout the entire image domain, the authors of [13] and [14] propose the following homogeneous diffusion inpainting scheme:

$$c(x)(u(x) - f(x)) + (1 - c(x))\Delta u(x) = 0, \quad (2.7)$$

with homogeneous Neumann boundary conditions implemented solely across the image boundaries that are encoded in the edge map (instead of also including the stored pixel values as Dirichlet boundary conditions). Here, x is a tuple indicating pixel location in a two-dimensional image, $c(x)$ is the value 0 or 1 depending on whether the stored pixel value is known at the location specified by x , u is the solution of the PDE, f is the original image, and Δu is the Laplacian of u . For simplicity, the two cases can be represented as follows:

$$\begin{cases} u(x) - f(x) = 0, & c(x) = 1 \\ \Delta u(x) = 0, & c(x) = 0 \end{cases} \quad (2.8)$$

The main difference of interest to us between the two papers is that [14] pursues the use of more advanced inpainting operators than the Laplacian. The two additional operators examined are the biharmonic operator and edge-enhancing diffusion (EED). The operator of more relevance to us is the EED operator. Inpainting across edges is an enduring issue which is why an exploration of this anisotropic nonlinear diffusion operator is worthwhile. In the EED formulation, Δu is replaced by $\text{div}(D(\nabla u_\sigma)\nabla u)$:

$$c(x)(u(x) - f(x)) + (1 - c(x))\text{div}(D(\nabla u_\sigma)\nabla u) = 0, \quad (2.9)$$

From calculus, we recall that in two dimensions the gradient $\nabla u = \left\langle \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right\rangle$ and the divergence of a vector field, $F = \langle F_1, F_2 \rangle$, is $\text{div}(F) = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y}$. Additionally, the positive definite matrix $D(\nabla u_\sigma)$ is an inhomogeneous diffusion tensor, where u_σ is the Gaussian smoothed version of the image u . The diffusion process is guided by the eigenvalues and eigenvectors of the diffusion tensor which depend on the gradient of u_σ . The eigenvectors are cleverly chosen so that one is orthogonal to ∇u_σ for full diffusion along image edges. The other is chosen to be parallel to ∇u_σ with an eigenvalue dependent on the contrast of neighboring pixels to reduce diffusion across high-contrast edges. Image inpainting with EED can reconstruct edges with high quality, even when the specified data is sparse [14].

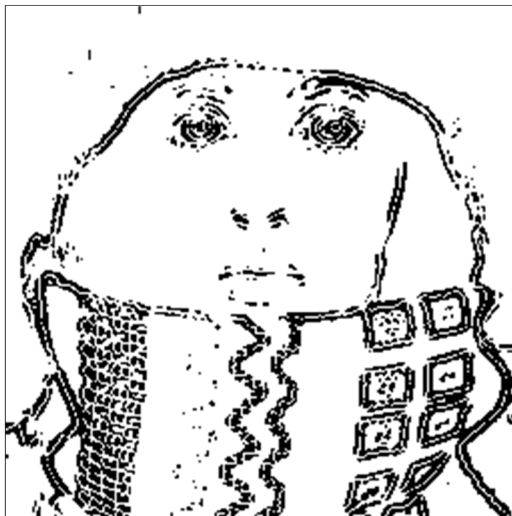


Figure 2-5 Inverted Laplacian magnitude after thresholding.

We were inspired by the appearance of the rescaled and inverted Laplacian magnitude due to the nature of the edges rather than the binary point mask. We immediately observed that this pre-dithering mask component had done a superb job of representing pixels on both sides of an edge. This preservation of pixel values and locations hinted at a solution to the blurred edges and patterns present in inpainted regions in our own work. Altering the pixel mask density allowed us to generate either more drastic or more subtle inverted Laplacian magnitude matrices (Figure 2-4, center), which were then utilized to create masks for damaged regions inpainting. Figure 2-5 shows a thresholded version of the inverse Laplacian magnitude in which the locations of values that are greater than $0.8 * \max(|\Delta I_G|^S)$ are represented. The lines on both sides of the contours can be visibly discerned. We explore how the Laplacian magnitude aids in damaged regions detection and edge preservation in image restoration in section 3.4.

3 Automatic Damaged Regions Detection Algorithm and Restoration of Damaged Photographs

Whether a digital image suffers from missing pixel information resulting from compression or transmission, or an old family photograph is damaged by age and is now cracked and discolored, digital inpainting techniques have tended to offer visually pleasing results. In the case of nondigital photographs, it can be difficult to design appropriate inpainting masks due to the nature of commonly encountered damage types. For example, old photographs may have damage manifest as scratches, blotches, or speckles that cover a large portion of the image. Although noise removal algorithms can help, it is often necessary to include an inpainting mask. Manually selecting damaged regions in a scanned photograph can be cumbersome and imprecise. In this chapter, we propose a solution to manual mask creation when the damage is difficult to encapsulate; a solution that is suitable for old photograph restoration.

3.1 Automatic Crack Detection in Photographed Scenes using Singular Value Decomposition

The authors of [17] use singular value decomposition (SVD) to automatically determine the locations of damaged regions in the scene of which the photograph was taken, thus eliminating the need for manual target region selection and mask creation. Their motivation is to allow viewers to visually appreciate the restored versions of the damaged objects without cracks and breaks associated with time and exposure to the elements. The authors use a sliding window

technique that compares two adjacent pixels based on m by n patches of neighboring pixels. For adjacent pixels, $p = (i, j)$ and $q = (i, j + 1)$ in two-dimensional image I , where $m = n = 3$, equations (3.1) and (3.2) represent the 3 by 3 patches associated with each pixel. Notice that there is a 3 by 2 patch of overlapping pixels. The column vectors v_p and v_q are the flattened versions of these patches as shown in equations (3.3) and (3.4). Once the 9 by 2 matrix A is constructed for patches ϕ_p and ϕ_q , singular value decomposition is applied to A . There are only two singular values in Σ on the diagonal and therefore, Σ can be reduced from a 9 by 2 matrix to a 2 by 2 matrix making it possible to find $V\Sigma$. Now, the similarity between the patches ϕ_p and ϕ_q is given by the cosine of the angle between w_1 and w_2 , where w_1 and w_2 are the rows of $V\Sigma$. Equation (3.6) shows this similarity measure.

$$\phi_p = \begin{bmatrix} I_{i,j} & I_{i,j+1} & I_{i,j+2} \\ I_{i+1,j} & I_{i+1,j+1} & I_{i+1,j+2} \\ I_{i+2,j} & I_{i+2,j+1} & I_{i+2,j+2} \end{bmatrix} \quad (3.1)$$

$$\phi_q = \begin{bmatrix} I_{i,j+1} & I_{i,j+2} & I_{i,j+3} \\ I_{i+1,j+1} & I_{i+1,j+2} & I_{i+1,j+3} \\ I_{i+2,j+1} & I_{i+2,j+2} & I_{i+2,j+3} \end{bmatrix} \quad (3.2)$$

$$v_p = [I_{i,j} \quad I_{i,j+1} \quad I_{i,j+2} \quad \dots \quad I_{i+2,j+2}]^T \quad (3.3)$$

$$v_q = [I_{i,j+1} \quad I_{i,j+2} \quad I_{i,j+3} \quad \dots \quad I_{i+2,j+3}]^T \quad (3.4)$$

$$A = [v_p \quad v_q] \quad (3.5)$$

$$\cos(\theta_{pq}) = \frac{w_1 \cdot w_2}{\|w_1\| \|w_2\|} \quad (3.6)$$

3.2 Our Improved Detection Method

We chose to use SVD to compare adjacent pixels but found that this primarily detected vertically damaged regions. We altered our program to detect horizontal and diagonal regions as well by including a third vector, w_3 , and adjusting the similarity metric. In our method, Σ is resized to be a 3 by 3 diagonal matrix with the singular values of A on the diagonal. Since $V\Sigma$ now has three rows, a third vector, w_3 , is introduced. The similarity measure used in our program is a modified version of equation (3.6) given by equation (3.10). Next, a 3 by 3 matrix is created for each primary pixel consisting of a subset of associated values calculated with $\cos(\theta_{pqr}^*)$. Finally, a similarity matrix of size I is produced where each element is calculated as a sum of all values contained within each subset of $\cos(\theta_{pqr}^*)$. The values of this similarity matrix are then compared with a threshold value δ , which is proportionally based on the size of ϕ_r .

$$\phi_r = \begin{bmatrix} I_{i+1,j} & I_{i+1,j+1} & I_{i+1,j+2} \\ I_{i+2,j} & I_{i+2,j+1} & I_{i+2,j+2} \\ I_{i+3,j} & I_{i+3,j+1} & I_{i+3,j+2} \end{bmatrix} \quad (3.7)$$

$$v_r = [I_{i+1,j} \quad I_{i+1,j+1} \quad I_{i+1,j+2} \quad \dots \quad I_{i+3,j+2}]^T \quad (3.8)$$

$$A = [v_p \quad v_q \quad v_r] \quad (3.9)$$

$$\cos(\theta_{pqr}^*) = \cos(\theta_{pq}) \cos(\theta_{qr}) \cos(\theta_{pr}) \quad (3.10)$$

Calculations were performed in MATLAB. Matrices U , V , and Σ were obtained using the *svd* function [18].

3.3 Experimental Results

We examined the usefulness of our proposed method considering both digital images of cracks in walls and pavement and scanned damaged photographs. Some of our results are presented in this section. Figure 3-1 shows a damaged painted walkway, the results from applying our damaged region detection algorithm with the output overlaid on the input image, and the results from using diffusion based inpainting with texture synthesis. The automatic detection results were generated with $m = n = 3$ and $\delta = \frac{1}{mn} * 0.99$. We utilized a technique similar to the hybrid structure and texture inpainting method proposed in [5] and described in section 2.2 of this dissertation.

After generating the inpainting mask from the automatic detection results, we implemented diffusion to fill the masked regions, then applied total variation denoising to the entire resulting image. This “denoised” image was created to capture the structural component, u , of the inpainted image, shown in part (c) of Figure 3-1. Subtracting the structural component from inpainted image produces the textural component, \hat{v} , shown in part (f) of Figure 3-1. To adopt the appropriate texture from the surrounding unmasked areas, we applied texture synthesis to \hat{v} , with a neighborhood size of 15 to obtain v , represented in part (e). We have rescaled and shifted \hat{v} and v so that they are easier to see.

Figure 3-2 also shows damaged pavement but with finer cracks than in Figure 3-1. The results from applying our automatic damaged regions detection algorithm with $m = n = 3$ and $\delta = \frac{1}{mn} * 0.94$ are overlaid on the input image, shown in part (b). We again utilized structure inpainting in tandem with texture synthesis. The result from using diffusion-based inpainting with texture synthesis is shown in part (d). After implementing diffusion to fill the masked

regions, we applied total variation denoising to the entire resulting image to capture the structural component, u , of the inpainted image, shown in part (c). Subtracting the structural component from inpainted image produces the textural component, \hat{v} , represented in part (f). Texture synthesis was applied to \hat{v} , with a neighborhood size of 15 to obtain v , represented in part (e). Again, \hat{v} and v have been rescaled and shifted.

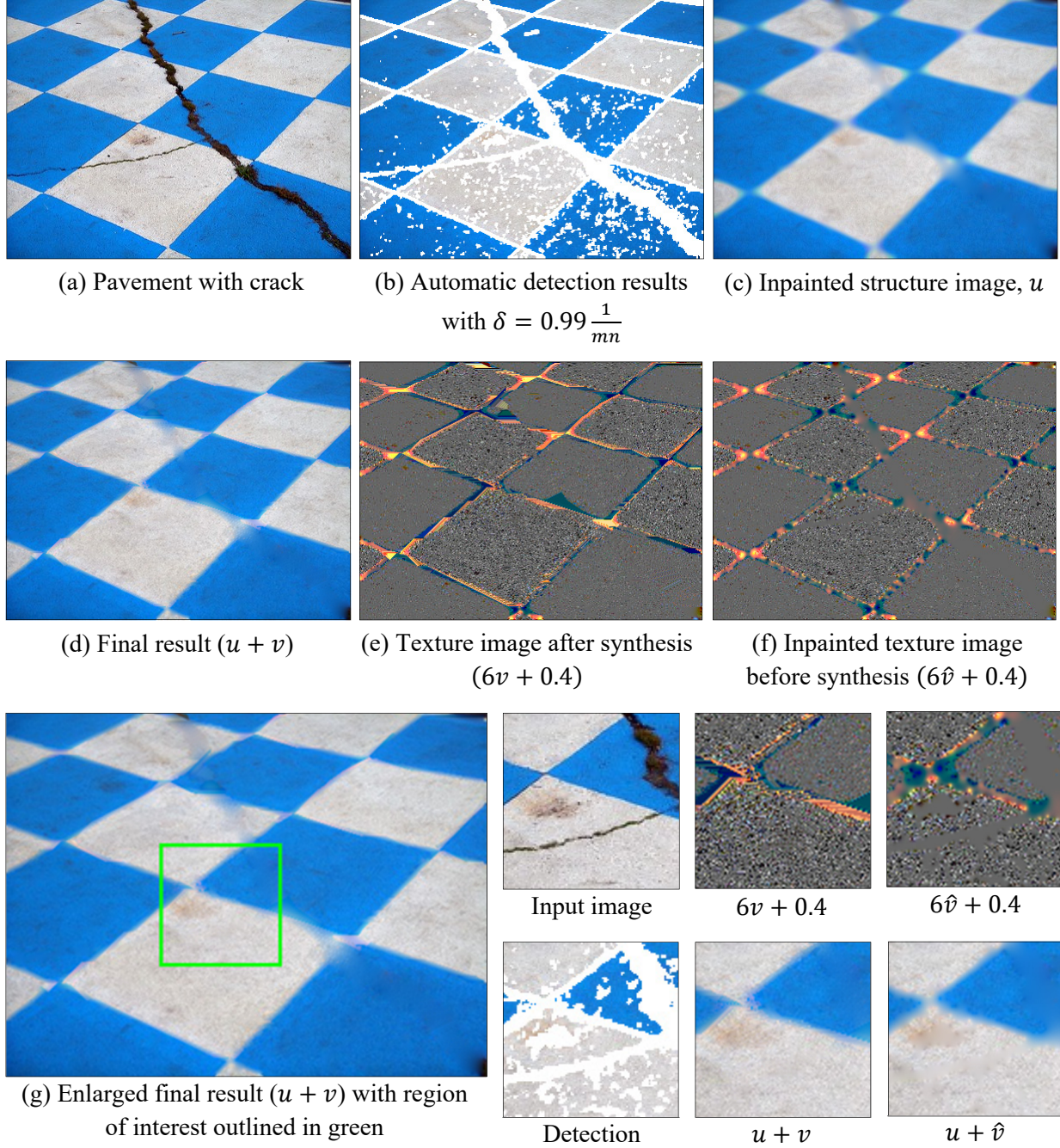


Figure 3-1 From left to right, top to bottom: (a) Image of cracked pavement¹ [19], (b) results from applying our automatic detection algorithm, (c) structure image after inpainting (b) with simple diffusion, (d) final result from adding the inpainted structure (c) and synthesized texture (e) images, (e) texture image after applying texture synthesis with neighborhood size of 15, (f) texture image before applying texture synthesis, (g) enlarged final result with selected region outlined in green and close-ups of the input image, the synthesized texture image, the

¹ A Crack in the Pavement of a Chequered Neighborhood by Matthew Rutledge is licensed under CC BY-NC 2.0. Resized from original.

pre-synthesized texture image, our detection results, our final results, and results without applying texture synthesis.

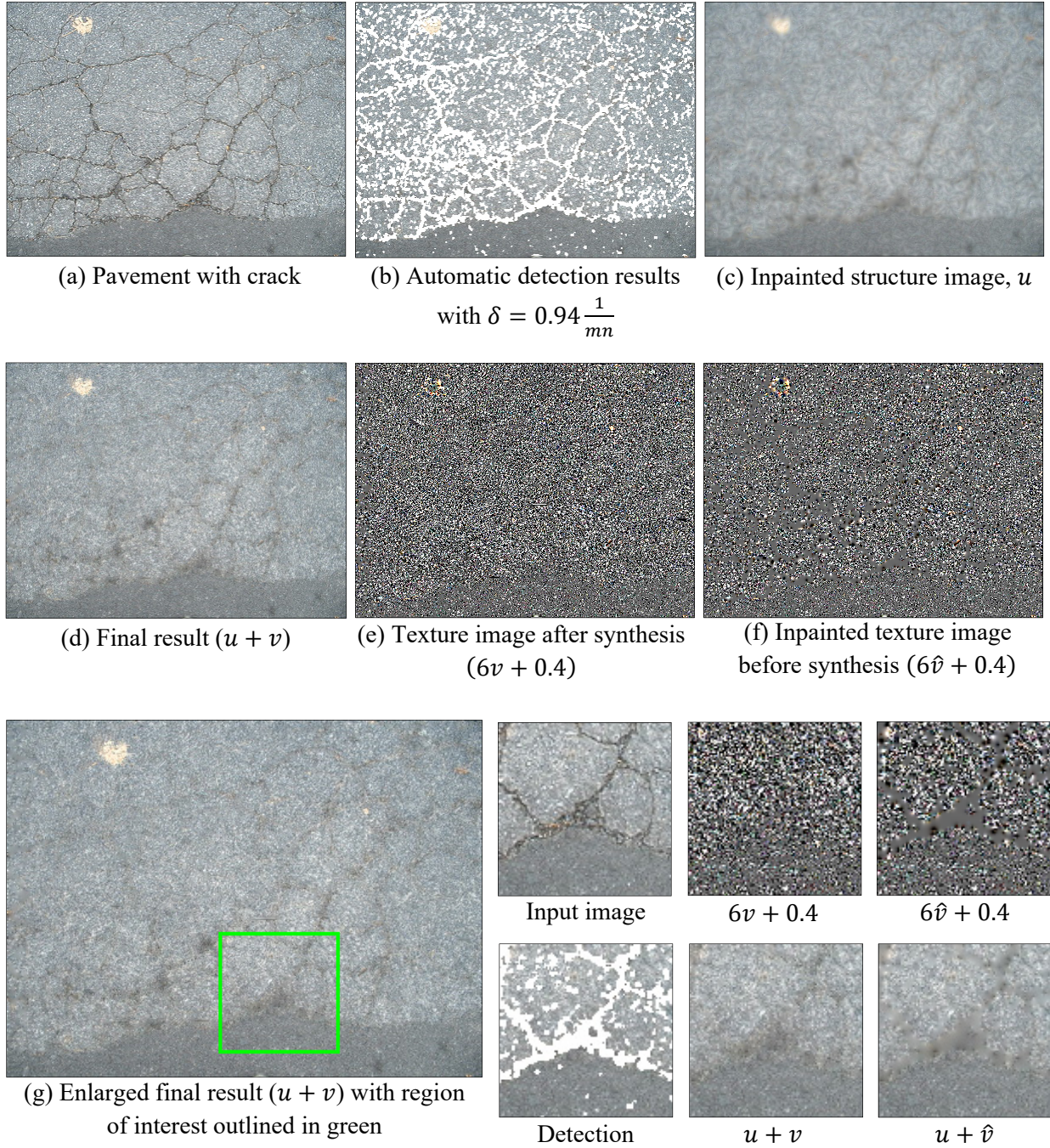


Figure 3-2 From left to right, top to bottom: (a) Image of cracked pavement² [20], (b) results from applying our automatic detection algorithm, (c) structure image after inpainting (b) with simple diffusion, (d) final result from adding the inpainted structure (c) and synthesized texture (e)

² Pavement cracks 1 (2) by alien_sunset is licensed under CC BY 2.0. Resized from original.

images, (e) texture image after applying texture synthesis with neighborhood size of 15, (f) texture image before applying texture synthesis, (g) enlarged final result with selected region outlined in green and close-ups of the input image, the synthesized texture image, the pre-synthesized texture image, our detection results, our final results, and results without applying texture synthesis.

Addressing damage at the border of the image is problematic because the size of ϕ restricts the algorithm from operating within a certain number of pixels from the edge. If ϕ is an $m \times n$ patch and the input image is $M \times N$, then the initial detection mask will be of shape $(M - m) \times (N - n)$. To superimpose the mask on the input image, we use padding so that the mask is the same size as the image. If m and n are odd values, there will be a line of ones of thickness $\frac{m-1}{2}$ pixels on the top edge, $\frac{n-1}{2}$ pixels on the left edge, $\frac{m+1}{2}$ pixels on the bottom edge, and $\frac{n+1}{2}$ pixels on the right edge. Our solution to this issue is to extend the ones-padded mask to the edge of the image using nearest neighbor mask values. For example, if the mask has a value of zero at the location $(M - 2, 2)$, indicating the location of a missing pixel in the bottom left corner of the input image, then the mask will be updated to contain zeros at locations $(M - 1, 2)$ and $(M, 2)$.

Once the image and mask are passed into the inpainting program, the issue again arises due to the nature of the diffusion operation which relies on intact neighboring pixels to calculate the missing pixel values. Because the damage is not rectangular, the diffusion operation is only applied if a mask value of 0 is reached. To inpaint damaged pixels at the very edges of an image, we applied the boundary condition equations outward toward the borders of the image after inpainting the interior of the image (rather than applying the boundary condition equations inward from the boundaries of the damaged regions before inpainting the interior).

Suppose I is the $M \times N$ input image to be repaired and B is the $M \times N$ binary mask in which a value of zero indicates the location of a missing or damaged pixel in I . For a 3×3 inpainting kernel, the one-pixel thick padded top border is inpainted as follows:

$$I(1, y) = \begin{cases} 2I(2, y) - I(3, y), & \text{if } B(1, y) = 0 \\ I(1, y), & \text{otherwise} \end{cases}, \quad (3.11)$$

where $y = 2:N - 2$. The operation begins at $y = 2$ because beginning at $y = 1$ would involve using the values of the missing pixels at $(2,1)$ and $(3,1)$ to inpaint the pixel at $(1,1)$. The pixel at $(1,1)$ is repaired when the one-pixel thick padded left border is inpainted. Furthermore, the operation ends at $y = N - 2$ since the right border has two-pixel thick padding and inpainting the pixel located at $(1, N - 1)$ using the values of the pixels at $(2, N - 1)$ and $(3, N - 1)$ would be counterproductive. The pixel at $(1, N - 1)$ is repaired when the two-pixel thick padded right border is inpainted.

The two-pixel thick padded bottom border is inpainted using

$$I(M - 1, y) = \begin{cases} 2I(M - 2, y) - I(M - 3, y), & \text{if } B(M - 1, y) = 0 \\ I(M - 1, y), & \text{otherwise} \end{cases}, \quad (3.12)$$

$$I(M, y) = \begin{cases} 2I(M - 1, y) - I(M - 2, y), & \text{if } B(M, y) = 0 \\ I(M, y), & \text{otherwise} \end{cases}, \quad (3.13)$$

where $y = 2:N - 2$. The one-pixel thick left border is inpainted with

$$I(x, 1) = \begin{cases} 2I(x, 2) - I(x, 3), & \text{if } B(x, 1) = 0 \\ I(x, 1), & \text{otherwise} \end{cases}, \quad (3.14)$$

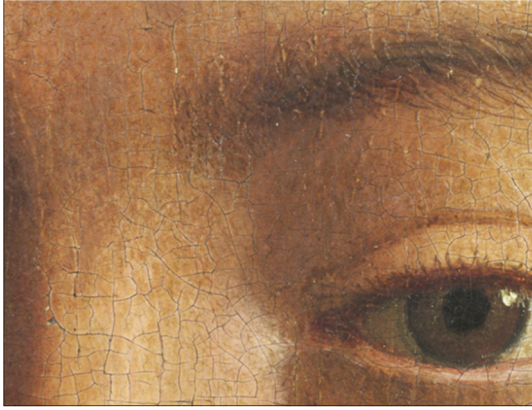
where $x = 1:M$. Utilizing the values at $(1,2)$ and $(1,3)$ to inpaint the missing pixel at $(1,1)$ is appropriate since each was inpainted when the one-pixel thick top border was repaired using (3.11). Lastly, the two-pixel thick padded right border is inpainted using

$$I(x, N - 1) = \begin{cases} 2I(x, N - 2) - I(x, N - 3), & \text{if } B(x, N - 1) = 0 \\ I(x, N - 1), & \text{otherwise} \end{cases}, \quad (3.15)$$

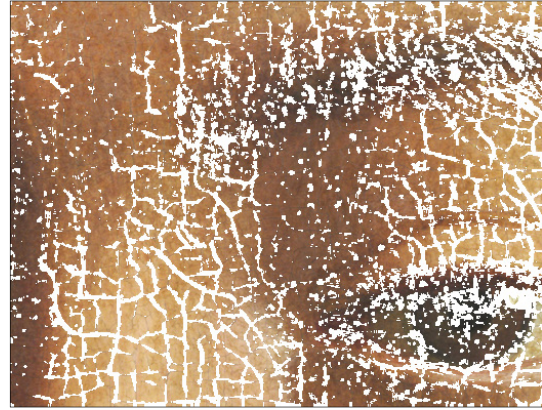
$$I(x, N) = \begin{cases} 2I(x, N - 1) - I(x, N - 2), & \text{if } B(x, N) = 0 \\ I(x, N), & \text{otherwise} \end{cases}, \quad (3.16)$$

where $x = 1:M$.

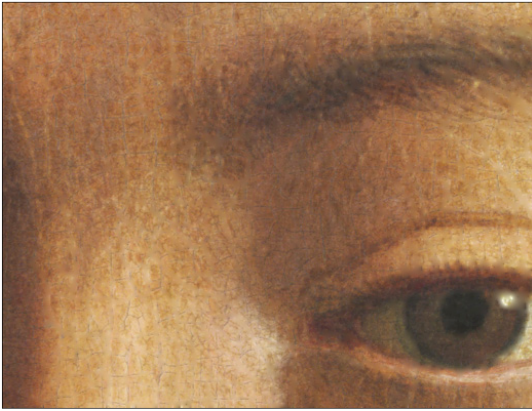
For images of pavement, resolving damaged regions at the exterior of the input image may seem trivial. However, in the case of artwork or portraiture, the ability to repair missing pixels even at the edges of the image is worthwhile. Figure 3-3 shows the results from applying our detection method and inpainting to the image of an old, cracked painting [21]. Without the use of SVD automatic detection, it would be tedious to manually create an inpainting mask. We have also included the results from using TV denoising and Gaussian smoothing, since these methods are considered reasonable initial attempts at restoration for images with fine, widespread damage as contained in Figure 3-3. Comparing the close-up images indicates that TV denoising does not diminish the appearance of the fine cracks whereas our method successfully detects many of the fine lines and fills in the missing information appropriately.



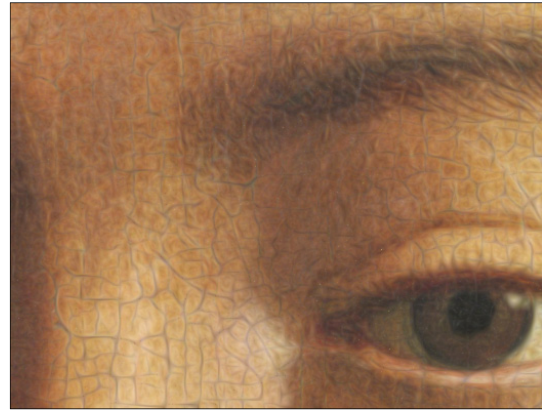
(a) Old oil painting with fine cracks



(b) Automatic detection results



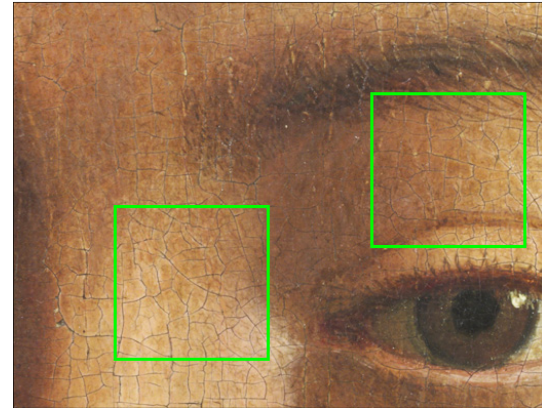
(c) Our restoration results



(d) TV denoising results



(e) Gaussian smoothing results, $\sigma = 2$



(f) Input image with ROIs outlined in green

Figure 3-3 From left to right, top to bottom: (a) Close-up of an old painting with very fine cracks [21], (b) automatic detection results using our SVD algorithm with $\delta = 0.991 \left(\frac{1}{mn} \right)$, (c) restoration results from applying inpainting, (d) restoration results from applying TV denoising to the entire image, (e) restoration results from applying isotropic Gaussian smoothing kernel with standard deviation of 2, and (f) same as (a) with two selected regions outlined in green to indicate the locations of the areas examined in Figure 3-4.

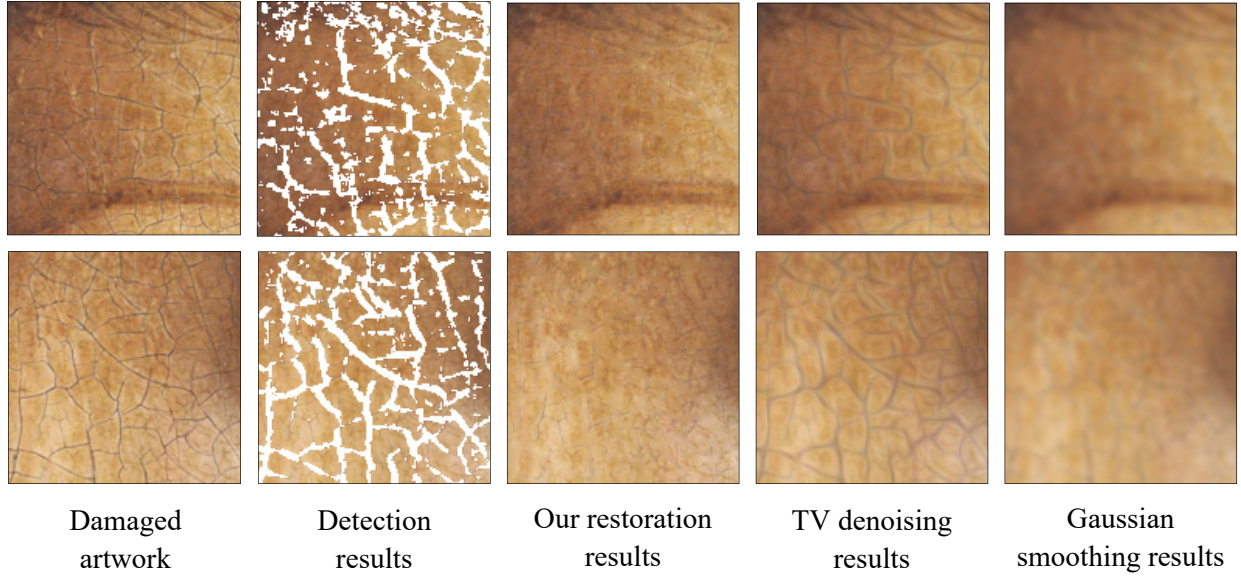


Figure 3-4 From left to right: Two regions of interest as established in part (f) of Figure 3-3 and their corresponding detection and restoration results, TV denoising results, and Gaussian smoothing results.

Figure 3-5 shows two 9×9 blocks taken from the same location in different restoration versions of the painting in Figure 3-4. The block on the left is taken from the inpainting results without applying boundary condition equations at the borders of the image. The block on the right is taken from the inpainting results achieved by utilizing boundary condition equations at the borders of the image. Notice that the pixels at indices (2,9) and (9,6) of the left block correspond to cracks that are resolved in the block on the right. The block is located at the bottom right of the input image where the diffusion that flows outward toward the bottom and right edges begins two pixels away from the edge.

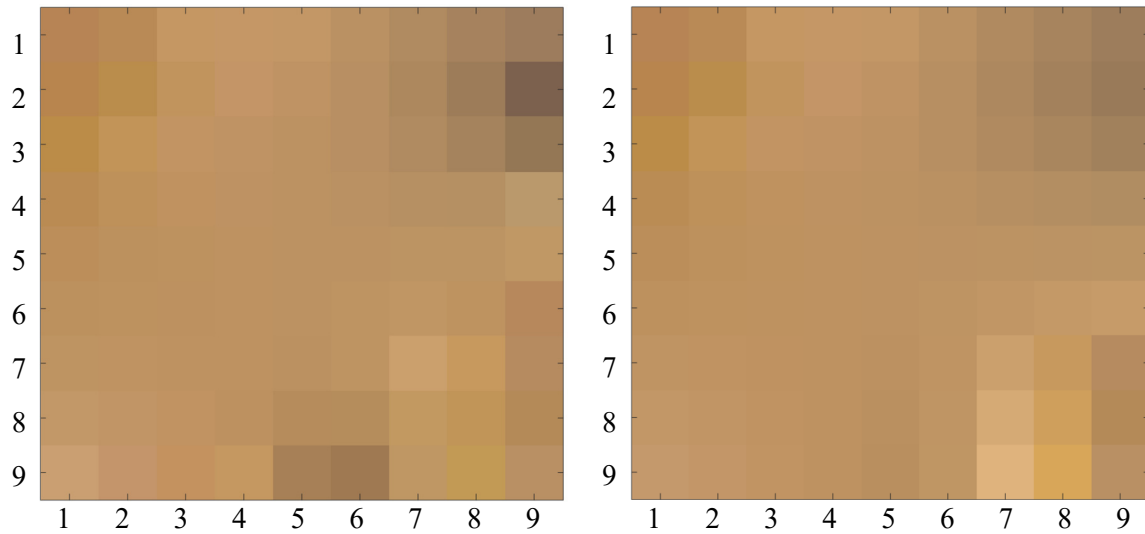


Figure 3-5 Two 9×9 blocks taken from the same location in different restoration versions of the painting in Figure 3-4. The block on the left is taken from the inpainting results without applying boundary condition equations at the borders of the image. The block on the right is taken from the inpainting results achieved with utilizing boundary condition equations at the borders of the image. Notice that the pixels at indices (2,9) and (9,6) of the left block correspond to cracks that are resolved in the block on the right. The block is located at the bottom right of the input image.

Graphic created in MATLAB with *imagesc* function [18].

Figure 3-6 shows a set of three scanned images of damaged photographs in the first column and the results from applying automatic damaged regions detection in the third column. To generate a mask that is thick enough to encompass the damaged regions, the program tends to over detect. This is particularly troublesome for images of people. Processing the damaged photographs in the same manner as was done in Figure 3-1 through Figure 3-3 does produce small visual improvements from the original scratched photographs, however, the blurred appearance around the eyes and mouths is unsuitable for portrait restoration.

Although not ideal, we experimented with manually adjusting the mask after running the automatic detection algorithm to prevent unfavorable results. By filling portions of the mask that merely outlined damage and removing parts of the mask that interfere with regions that should remain intact, the restoration results are much more appropriate. The last column of Figure 3-6

shows the tweaked mask with the inpainting results in the second column. Although this type of intervention seems unsuitable for a method that seeks to limit manual mask creation, it does inspire potentially automatic algorithms. Also, we found that in cases where manual mask creation is necessary, it makes preparing the inpainting mask faster and more precise. We believe that with more experimental exploration and mathematical thoughtfulness a successful hybrid method that further diminishes manual detection is possible for many different types of images containing damage that is widespread or difficult to represent.



Figure 3-6 Three sets of images corresponding to three damaged photographs³ [22] [23] [24]. From left to right: Input damaged photograph, restoration results from using the partially user dependent method of mask creation method, automatic damaged regions detection results, and modified mask which is used to inpaint the damaged photograph in the first column.

³ *Torn Victorian Photo Pre-restoration* provided by John Butler of PhotoVale is used with permission. Cropped and resized from original.

Photo booth portrait of a chubby man 2: Dad 1940 by simpleinsomnia is licensed under CC BY 2.0. Cropped and resized from original.

3.4 Utilizing the Inverted Laplacian Magnitude for Image Compression in Damaged Regions Inpainting Tasks

In this chapter, inpainting tasks entail detecting damage and creating a mask based on the location of this damage. The mask indicates where the inpainting algorithm should be applied within the damaged image. When an image is impervious to this detection method because it is splotched and dotted, we borrow a key tool from the compression methods examined in section 2.4. The techniques discussed in [13] and [14] employ inpainting with homogeneous diffusion and utilize the inverted Laplacian magnitude in their compression algorithm. An image that is a candidate for combining our detection and inpainting goals with the image decoding inpainting methods suggested in [13] and [14] is shown in Figure 3-7 [25]. The reason for this is because much of the damage is characterized by dots and splotches, rather than creases or tears. Zooming in on damage that appears line-like shows that the lines are comprised of dots, Figure 3-8. Nonetheless, there are damaged portions of the image that are comprised of lines. When this is the case, our proposed automatic SVD-based method of inpainting mask creation fails as can be seen in Figure 3-9.



Figure 3-7 Image with damage that is characterized by light colored spots throughout and concentrated over the face of the subject [25].



Figure 3-8 Input image with selected region inscribed in a red square and the corresponding close-up revealing that the damage is characterized by dots and splotching.

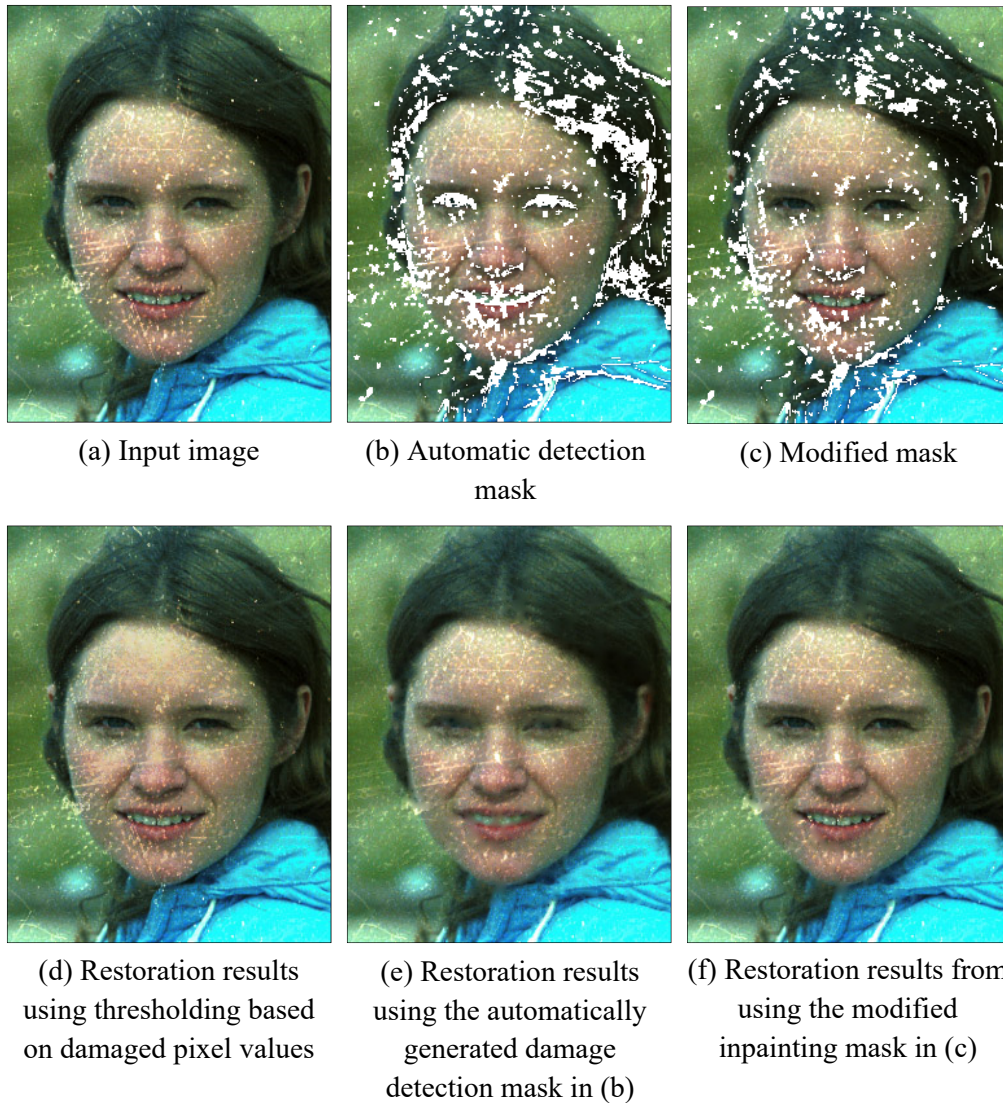


Figure 3-9 SVD-based automatic detection and inpainting results. Notice the blurred edge of the girl's right cheek, chin, and eyes in part (e).

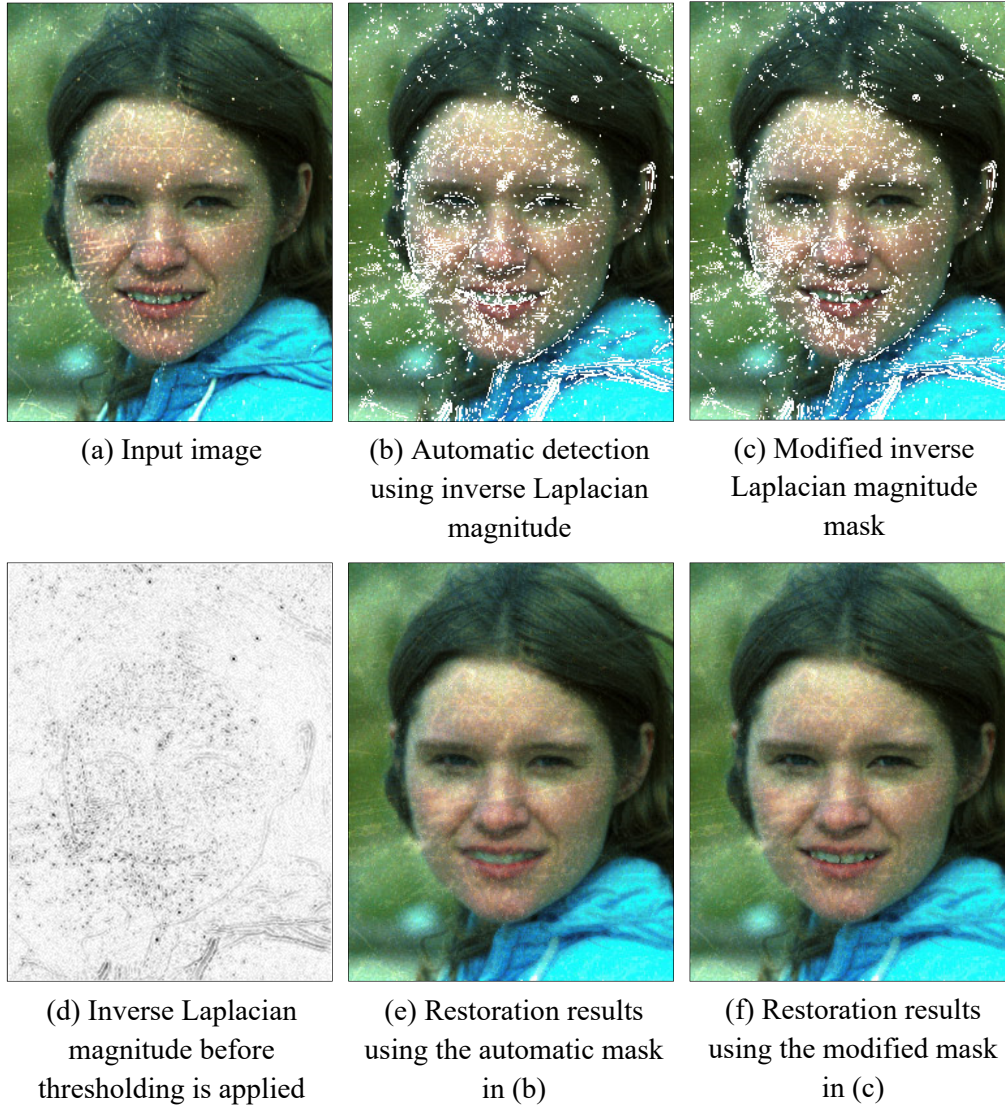


Figure 3-10 Inverse Laplacian magnitude automatic detection mask results. The edges of the girl's right cheek, chin, and eyes are preserved.

Unlike the authors of [14] and [13], we do not have access to the original image, referred to as f in [14], in our work and therefore we are unable to use the same mean square error, MSE, approach for discerning reconstruction error. We have therefore limited our analysis to visual improvements. The inpainting mask that we created using the Laplacian magnitude did a good job of preserving edges in the original image although it did cause blurring in regions that were

not damaged. We believe the successful preservation of edges is due to the way the threshold inpainting mask straddles both sides of an edge. Figure 3-11 (a) shows a magnified view, right, of the inpainting mask on the left. Part (b) shows the same selected region but with the results from applying Sobel edge detection to the damaged input image in yellow. The double striping can be readily observed, particularly where the girl's right cheek fronts her hair and the image background, even amidst heavy damage.

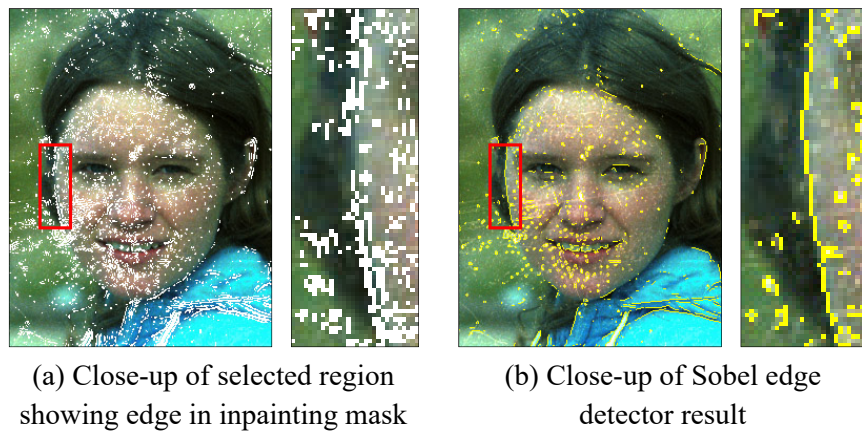


Figure 3-11 Demonstration of how the Laplacian magnitude mask creates lines that straddle the edges rather than directly on top of the edges. Sobel edge detection in the image on the right indicates the location of the edge that the mask in the image on the left outlines.

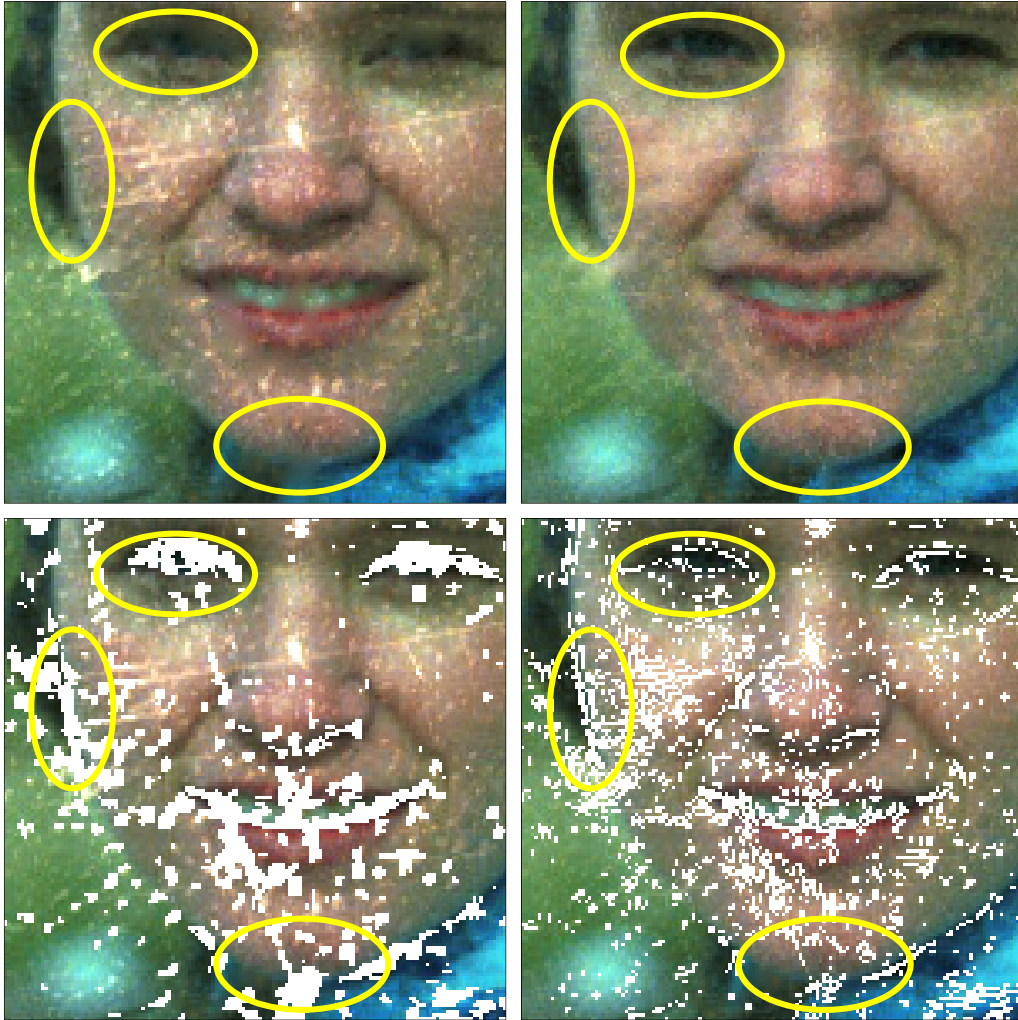


Figure 3-12 Raw results from using the SVD detection and inverted Laplacian magnitude detection (without mask modifications). The two top images demonstrate the difference between the two detection methods. The top image on the left shows blurring in areas where edges exist and should be preserved whereas the top right image shows edges remain uncompromised. The two bottom images are the inpainting masks detected using SVD and Laplacian magnitude, respectively. Zoom in for better viewing.

There is certainly a compromise between image quality and wholly automatic damaged regions detection. For example, in order to successfully detect the damaged regions in an old photograph without user intervention, the threshold for comparison may be such that the algorithm over-detects. This means that although the program capably locates the regions in need of inpainting, it may also include non-damaged areas. Since we use the automatically detected regions as inpainting masks when implementing restoration, the repaired image may appear

foggy or grainy in places that were previously clear. This issue occurs commonly around the eyes in images of people. In the future, we would like to create a program, that successfully identifies damaged regions, produces a mask, and applies inpainting techniques with minimal human intervention.

3.5 Conclusion

After researching and experimenting with digital color image inpainting techniques, we developed the foundations of an automatic damaged regions detection algorithm. Desiring to find a technique that performs well independent of user intervention, we utilized singular value decomposition with the reduction suggested in [17]. Once the damaged region is detected, the damaged image and the detected region matrix are passed into a function that treats the detected region as a mask and applies diffusion based inpainting. We experimented with various diffusion and TV parameters, patch sizes, and similarity thresholds. We built a program capable of detecting horizontal, vertical, and diagonal damage, and utilized the Laplacian magnitude of the input image to pinpoint where edge preservation is necessary based on the compression methods in [13] and [14]. We then applied inpainting techniques accordingly. Although there is plenty of work being done in the field of non-machine learning-based inpainting, we would like to continue exploring image restoration techniques, particularly those that do not involve or require user intervention.

4 Machine Learning-Based Image Restoration

Recent machine learning image restoration approaches tend to favor either generative adversarial networks (GANs) [26] [27] [28] [29] [30] [31] or encoder-decoder CNNs [32] [33] [34] with residual learning. GANs train two networks simultaneously; a generative model and a discriminative model. The goal of the generative model is to generate samples that are indistinguishable from the training set and the goal of the discriminative model is to correctly determine if the sample came from the generative model or the training set. Their adversarial relationship ensures that the generator produces images that are globally and locally believable so that the discriminator fails, meanwhile, the discriminator becomes pickier to overcome the continually improving generator output images.

Encoder-decoders are two-step models similar to autoencoders. Just as an autoencoder tries to reconstruct its input, encoder-decoders can be used to generate an output that is nearly identical to its input. In image restoration, the goal is to train an encoder-decoder to output the input image sans the missing or noisy regions. It has been shown that utilizing skip connections in an encoder-decoder network is beneficial in segmentation and inpainting tasks [3] [32] [33] [34]. Skip connections concatenate weight maps produced in the encoder stage to their mirrored counterpart weight maps in the decoder stage. This encourages preservation of local features and structures from the damaged image in the predicted image. In response to general image-to-image translation problems, in which the input and output have the same underlying structure,

[35] employs conditional GANs and encoder-decoders by designing a U-Net-like generator that uses skip connections. We will discuss the U-Net in detail in chapter 5.

4.1 Loss

As mentioned earlier, image-to-image translation tasks often require that the input and output images be structurally identical. Image super resolution, style transfer, and restoration are examples of image-to-image translation that further require similarity in features like color, texture, and contour. Training a CNN with per-pixel loss may improve objective metric values, such as peak signal to noise ratio (PSNR) and ℓ_1 and ℓ_2 losses yet fail to produce perceptually satisfactory results. In image inpainting, the goal is to restore missing or damaged regions, of which there could be many possible suitable solutions. Inpainting is therefore an ill-posed problem and it can be difficult to measure success using objective metrics. Fortunately, there has been much research within the last five years to mathematically incorporate features that are perceptually significant to the human visual system (HVS) [36] [37].

Perceptual quality can be broken down into two components: content and style. Image content representation describes the global features of an image, whereas style representation describes the local features of an image such as texture and color. By examining feature responses in certain layers of a CNN trained for object recognition, Gatys et. al. determined that content and style image representations are independent of each other and therefore can be managed separately [36]. The specific CNN used in their work is the highly successful VGG19 network [38] trained on the ImageNet dataset [39] for object recognition. High level features of the original image and the predicted image are obtained by exposing each one to selected layers of the VGG19 network.

For content reconstruction, the goal is to minimize the difference between the two feature response maps generated by the original image and the predicted image. Using standard error back propagation, the updated predicted image will eventually take on values that trigger the same feature responses as the original image. Given a pretrained CNN, ϕ , the content loss associated with original (input) image I and predicted (output) image P at layer l of ϕ is

$$L_{content}(I, P, l) = \frac{1}{mnc} \|\phi_l(P) - \phi_l(I)\|_2, \quad (4.1)$$

where $m \times n$ is the shape of the feature map, c is the number of feature maps (or channels) in layer l , and $\phi_l(P)$ and $\phi_l(I)$ are the feature responses from processing P and I , respectively.

Image style representation is described by the correlations between the feature responses in the same layer [36]. For an image I , these correlations are obtained by finding the Gram matrix of the feature responses, $\phi_l(I)$, in a specific layer l of network ϕ . To calculate the style loss associated with the predicted image P , the set of feature responses, $\phi_l(P)$, is reshaped from an $m \times n \times c$ matrix to a $c \times mn$ matrix, $\psi_l(P)$. Thus, the Gram matrix,

$$G_l(P) = \frac{1}{mnc} \psi_l(P)^T \psi_l(P), \quad (4.2)$$

will be of shape $c \times c$ and the style loss associated with original image I and predicted image P at layer l of ϕ is

$$L_{style}(I, P, l) = \frac{1}{c^2} \|G_l(P) - G_l(I)\|_2. \quad (4.3)$$

As mentioned in section 2.3, TV regularization is used to suppress noise while preserving edges in noisy images. Unsurprisingly, TV loss is used in image-to-image translation tasks to produce smooth predicted images. The TV loss for the predicted image P is

$$L_{tv}(P) = \frac{1}{mnc} \sum_{ijk} \sqrt{(P_{i+1,j} - P_{i,j})^2 + (P_{i,j+1} - P_{i,j})^2}. \quad (4.4)$$

The ground truth image is not included in the calculation since TV loss solely penalizes the amount of noise in the generated image with no comparison to the original image. We utilize TV loss in our region hiding inpainting system to compare the boundary of the inpainted regions with the one-pixel thick set of valid pixels surrounding the inpainted region.

5 Region Hiding for Image Inpainting via Single-Image Training of U-Net

5.1 Introduction

Inspired by the impressive performance of CNNs in inpainting tasks coupled with an awareness of the inherent difficulties associated with the creation and management of a large data set of images, we desired to explore how a specific encoder-decoder CNN, the U-Net [3], would perform on inpainting tasks when trained on a single image; the very image to be inpainted. Our single-image trained model not only performs well when compared to multi-image trained inpainting systems, but also lessens reliance on large datasets of possibly unsuitable images. The results also indicate how effective the combination of U-Net with partial convolutional layers and skip connections is [34]. We will refer to the network architecture proposed in [34], which we use in our work, as partial convolutional neural network (PConvNN).

What makes our inpainting system unique is that it relies solely on one image rather than a database of images. Training is exceptionally fast compared to other machine learning inpainting systems that have similar architecture but train on thousands of images [32] [33] [34]. Although a fair method-to-method comparison is not possible, ours being the only machine learning system that we are aware of to use solely one image for training, we have validated our region hiding method by comparing our results with the results generated from applying a pretrained, third-party implementation of [34], which was trained on the ImageNet data set [39], and a third-party implementation of PatchMatch [10], to the same set of target images [40] [41] [42] [43] [44]. The ImageNet data set contains over fourteen million images from a variety of

categories. This makes the associated pretrained model an appropriate choice for comparing our single-image trained PConvNN with the multi-image trained PConvNN.

In this chapter, we explain what training on a single image using region hiding means, the model architecture used, how we went about implementing the model, and our results. Lastly, we suggest potential applications and discuss future research stemming from this work.

5.2 Approach

The novel contribution that we describe in this paper is conceptual, rather than computational, although it does affect computational efficacy and will undoubtedly be of value in inpainting tasks as well as in other signal processing domains. We do not claim to offer a new implementation. Rather, we have thoughtfully leveraged a successful, previously proposed multi-image inpainting network [34] to accomplish our inpainting goals pertaining to single images. In this section, we describe how and why various components are used, including architecture.

5.2.1 Region Hiding

We refer to our novel contribution as region hiding because the algorithm hides nondamaged regions from the network. Region hiding enforces that the network be trained to make predictions based on what it sees in the current training image, as well as what it has seen in previous training images, i.e., in the various transformations of the damaged image. Nuances associated with the structures, textures, and overall cohesion of elements within that specific image are learned. The network is not rewarded for using damaged pixels to predict hidden regions.

5.2.2 Set-Up

There are three important component images that correspond to a test image. First, the ground truth image without damage or masking, I_0 . Second, the damaged version of I_0 , which has had damage artificially applied. We will call this damaged image I . A sample set of I_0 and I is shown in the first two images of Figure 5-1. Third, the masked version of I , which we will call I_M . There is one I_0 and one I , however, there will be several versions of I_M depending on augmentation specifications. A set of I_M s corresponding to the sample in Figure 5-1 is shown in the last column of Figure 5-2.



Figure 5-1 From left to right, ground truth image⁴ (I_0), damaged image (I) with damage shown as white, and binary mask representing locations of artificially damaged pixels in black.

⁴ *Color Block Living* by Sheila Sund MD is licensed under CC BY 2.0. Cropped and resized from original.

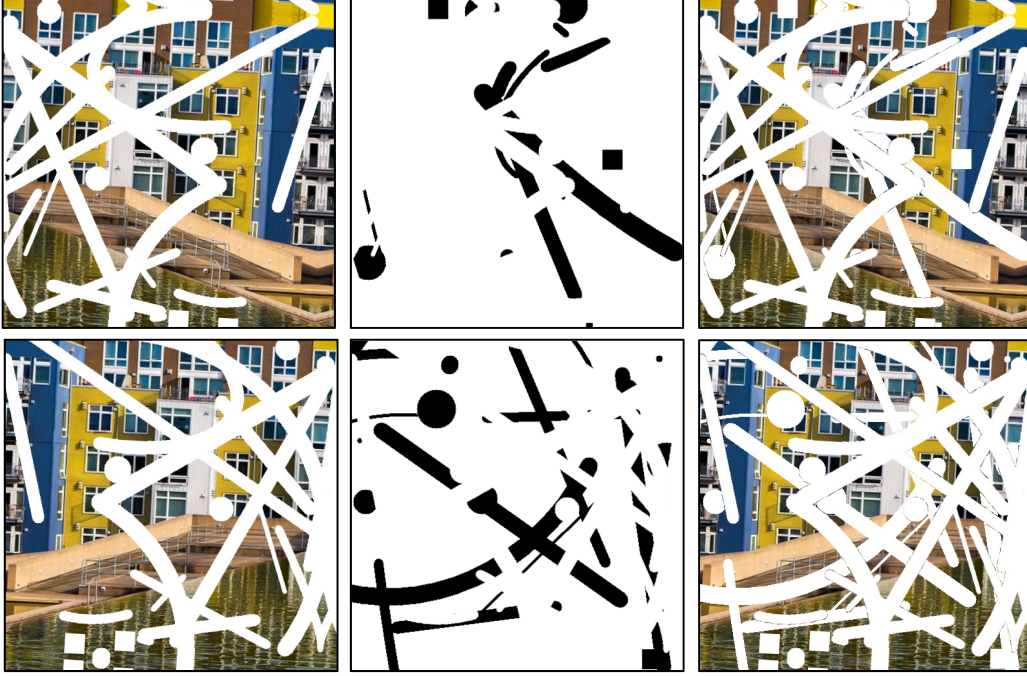


Figure 5-2 Two instances of transformation and mask generation. From left to right, transformed damaged image, mask such that damage is not included, and transformed damaged image with mask applied (I_M).

GANs are not suited for single-image inpainting since both the generator and the discriminator must compete while handicapped by identical masked damaged images. A GAN that is trained on a single image will insist that the damage is part of the signal, not noise, because the generator has been repeatedly awarded by the discriminator for maintaining the damaged regions in its predictions. An encoder-decoder system is ideal for single-image inpainting because precise mask placement can be used to prevent the model from actively making predictions about the damaged region until the testing phase.

Although we could not find publications pertaining to single-image training in machine learning-based inpainting, the authors of [45] likewise do not train on a large dataset of images. However, they take this notion a step further and do not train on the damaged image either. Our method differs in that we do indeed train on an image, as we do not set network parameter priors. Our method encourages the network to learn the act of inpainting as it pertains to a specific

image. Nonetheless, we draw the same conclusion that CNNs are powerful tools for image restoration due to their architecture, independent of training on large image datasets.

We chose to adopt the architecture proposed in [34] because it offers solutions to challenges commonly encountered in both traditional and machine learning-based inpainting approaches. Specifically, the system is capable of training on images with irregular masks and placeholder values are not required to explicitly inform the model where to inpaint. The novel contribution proposed in [34] is the development and implementation of partial convolutions to fill masked regions. The inspiration for utilizing masked convolutions and renormalization comes from segmentation-aware convolutional networks [46]. The authors of [34] refer to the combination of segmentation-aware convolution [46] and automatic mask updating as the partial convolutional layer.

Let $F_l(i)$ be a $k \times k$ receptive field centered at position i in the feature map F_l in layer l and let $M_l(i)$ be the associated binary mask of the same size in which a value of 0 indicates a masked pixel and a value of 1 indicates an unmasked (unhidden) pixel. The output from performing partial convolution at this receptive field is,

$$f_{l+1}(i') = \begin{cases} W^T(F_l(i) \odot M_l(i)) \frac{k^2}{\sum M_l(i)} + b, & \text{if } \sum M_l(i) > 0, \\ 0, & \text{otherwise} \end{cases}, \quad (5.1)$$

where \odot indicates elementwise multiplication (Hadamard product), $\sum M_l(i)$ is the number of unhidden pixels in the receptive field, W is the weight matrix, and b is the bias term for the feature map. The left side of (5.1), $f_{l+1}(i')$, is the raw output from the convolution. The resulting value from applying the activation function to this raw output is assigned to the neuron located at

i' of the feature map F_{l+1} in layer $l + 1$. The partial convolutional layer is complete after the mask updating step,

$$m_{l+1}(i') = \begin{cases} 1, & \text{if } \sum M_l(i) > 0 \\ 0, & \text{otherwise} \end{cases}, \quad (5.2)$$

where $m_{l+1}(i')$ will be assigned a value of 1 if the calculation of $f_{l+1}(i')$ is based on at least one unhidden input, therefore implying that there is now a valid pixel value at location i' in F_{l+1} . The updated mask travels with the updated image throughout the network. The number of masked pixels will eventually decrease to zero and the mask will contain all ones by the time the bottle neck between the encoder and the decoder is reached, assuming that the input image contains at least some unhidden pixels.

5.2.3 Data Augmentation

The U-Net is an increasingly popular encoder-decoder CNN that has been highly successful in image segmentation and inpainting. U-Net is often employed in projects suffering from a small number of training images. Initially, U-Net was introduced as a robust, fully convolutional network designed for various biomedical image segmentation tasks [3]. Each task discussed in [3] requires the network to train on very few images; one of which contains a dataset with a mere 20 training images. The authors implement elastic deformations to augment the available dataset, thus encouraging the network to become familiar with deformations that are possible in tissues and neuronal structures which may not be explicitly present in the training set. Given that we are training on a single image, data augmentation is especially important.

We utilize rotations, shifts, and horizontal flips applied to the damaged input image to impose diversity in the training images. Keras [47] offers a helpful set of tools to facilitate real-

time data augmentation in its *ImageDataGenerator* class. Although a true application of the inpainting system would imply that the input image has been genuinely damaged in a way that the missing pixel values are unknown, we apply artificial damage to the original, ground truth image, so that we can accurately measure success. To preserve the locations of these artificially damaged regions from one training image to the next, both the damaged image and the damage are transformed simultaneously. Next, a random mask that excludes the damaged pixel locations is applied to each training image as shown in Figure 5-2.

There is no overlap between the mask and damage. When the mask is subjected to the mask updating step, the damage is not included. The model is designed to predict the pixel values of the hidden regions distinct from the way that the model predicts the pixel values of the unmasked regions (which includes damage). This purposeful bias is accomplished in the loss function.

5.2.4 Loss

Rather than determine a unique set of loss term coefficients for each image that needs to be inpainted, we used the loss function described in [34], which the authors obtained by performing a hyperparameter search on 100 validation images taken from the ImageNet, Places2, and CelebA-HQ datasets. The authors use an effective total loss function that embodies pixel loss within and outside of the masked regions (L_{hidden} and L_{valid} , respectively), perceptual and style losses [36] ($L_{perceptual}$ and L_{style}) dependent on ImageNet-trained VGG16 [38] feature space, and total variation loss [37], L_{tv} . Although we present the ℓ_2 losses for content and style in equations (4.1) and (4.3), in section 4.1, in our work each loss term is calculated as an ℓ_1 loss as in [34].

$$L_{total} = a_1 L_{valid} + a_2 L_{hidden} + a_3 L_{perceptual} + a_4 L_{style} + a_5 L_{tv}. \quad (5.3)$$

Here, the a_i s are the weights associated with each loss term.

We discovered that although perceptual and style losses do not noticeably improve the appearance of predicted images for some highly successful inpainting networks such as [29], they are indeed essential components of our loss function. The importance of these loss terms can be qualitatively appreciated in section 5.5.3, which includes ablation study findings.

5.2.5 Architecture

The name of the U-Net comes from its symmetrical U-shaped architecture. The U-Net-like PConvNN consists of an encoder stage and a decoder stage, each of which is comprised of stacked partial convolutional layers. The $m \times n \times 3$ masked damaged image, I_M , is input into the network which starts with a convolutional layer that uses either zero padding or partial convolution based padding [48] and horizontal and vertical strides of size 2, to output a collection of c feature maps that are of reduced length and width. This process continues until a bottle neck is reached at which point the decoder stage begins. Skip connections bridge mirrored layers by concatenating encoder stage output channels to decoder stage input channels after nearest neighbor upsampling. This implies that the network will model $g(I) = h(I) - I$, where g is the network function into which the damaged image, I , is input, and $h(I) = I_0 + I$, ideally. The last layer of the PConvNN concatenates the network input to the output of the second to last layer, applies the last partial convolution layer, applies sigmoid activation, then outputs the result. Table 5-1 shows the architecture of the PConvNN that we employed. We have included a schematic representation in Appendix D.

Table 5-1: Partial Convolutional Network Architecture

Layer	Type	Layer output maps (count and size)		Kernel size	Stride	Upsample	Activation
In	Input	3	512×512	--	--	--	--
1	PConvolution	64	256×256	7×7	2	--	ReLU
2	PConvolution	128	128×128	5×5	2	--	ReLU
3	PConvolution	256	64×64	5×5	2	--	ReLU
4	PConvolution	512	32×32	3×3	2	--	ReLU
5	PConvolution	512	16×16	3×3	2	--	ReLU
6	PConvolution	512	8×8	3×3	2	--	ReLU
7	PConvolution	512	4×4	3×3	2	--	ReLU
8	PConvolution	512	2×2	3×3	2	--	ReLU
9	Connect to 7	1024	4×4		--	2	LeakyReLU(0.2)
	PConvolution	512	4×4	3×3	1	--	
10	Connect to 6	1024	8×8		--	2	LeakyReLU(0.2)
	PConvolution	512	8×8	3×3	1	--	
11	Connect to 5	1024	16×16		--	2	LeakyReLU(0.2)
	PConvolution	512	16×16	3×3	1	--	
12	Connect to 4	1024	32×32		--	2	LeakyReLU(0.2)
	PConvolution	512	32×32	3×3	1	--	
13	Connect to 3	768	64×64		--	2	LeakyReLU(0.2)
	PConvolution	256	64×64	3×3	1	--	
14	Connect to 2	384	128×128		--	2	LeakyReLU(0.2)
	PConvolution	128	128×128	3×3	1	--	
15	Connect to 1	192	256×256		--	2	LeakyReLU(0.2)
	PConvolution	64	256×256	3×3	1	--	
16 Out	Connect to In	67	512×512		--	2	Sigmoid
	PConvolution	3	512×512	3×3	1	--	

5.3 Implementation

Like [34], we use He uniform weight initialization [49] and LeakyReLU with $\alpha = 0.2$ for the nonlinear activation function. We also scale partial convolutional layer outputs at individual

receptive field (sliding window) locations similarly by multiplying the convolution output by the ratio of the total number of pixels in the receptive field to the number of unhidden pixels in the receptive field as represented in (5.1). If there are many masked pixels in the receptive field, then the ratio will be large to compensate for the few nonzero values in $F_l(i) \odot M_l(i)$, and if there are many unhidden pixels, then the ratio will appropriately be closer to one. This ratio prevents distortion resulting from high variance among values in the same feature map.

In practice, we found that an alternative scaling factor based on the size of the entire mask in layer l and the number of nonzero pixels in the entire mask in layer l , generated more structurally cohesive results. Changing the locally calculated ratio in (5.1) to this globally calculated scaling factor gives us:

$$f_{l+1}(i') = \begin{cases} W^T(F_l(i) \odot M_l(i)) \frac{mn}{\sum M_l} + b, & \text{if } \sum M_l > 0 \\ 0, & \text{otherwise} \end{cases}, \quad (5.4)$$

where $m \times n$ is the size of the mask in layer l and $\sum M_l$ is the sum of all pixels in the mask in layer l , i.e., the number of unhidden pixels in the feature map, F_l . Rather than dividing the size of the receptive field centered on i by the number of unmasked pixels in the receptive field, we now divide the size of the entire mask in layer l by the number of unmasked pixels in the entire feature map in layer l . We provide the results from using both scaling factors in section 5.5.

Batch normalization is unnecessary for the encoder stage of our network since a batch consists of shifts and rotations of the same damaged image with varying amounts of hidden pixels. Applying normalization would produce skewed values for the empirical mean and standard deviation associated with an individual batch of images and later influence the intensity

of the predicted image. However, batch normalization is used in the decoder stage as we found that images with heavy damage tended to produce bright images. To prevent bright image predictions, we assigned the mean of the nondamaged pixels to the damaged regions as described in (5.5) through (5.7).

There are two types of masks to be aware of; the artificially applied damage mask, D , and the hidden region mask, M . The damage mask is what is used to create the damaged image, I . When D is generated, the damaged regions are assigned a value of 0 and the nondamaged regions are assigned a value of 1. An intermediate image, \hat{I} , is created by elementwise multiplication of the damage mask and the original image:

$$\hat{I} = D \odot I_0. \quad (5.5)$$

The number of damaged pixels in a channel is calculated by subtracting the number of nondamaged pixels from the total number of pixels:

$$d = mn - \frac{\sum D}{3}, \quad (5.6)$$

where m and n are the length and width of the input image and $\sum D$ is the sum of all pixel values in the damage mask, i.e., the number of nondamaged pixels. This sum is divided by 3 since there are 3 color channels and we are interested in finding the mean of the nondamaged pixels for each color channel independently.

The mean of all nondamaged pixels in a specific color channel is calculated by dividing the sum of all nondamaged pixel values by the number of nondamaged pixels. This is represented in (5.7).

$$I_C(i) = \begin{cases} \frac{\sum \hat{I}_C - 255d}{mn - d}, & \text{if } D_C(i) = 0 \\ \hat{I}_C(i), & \text{otherwise} \end{cases} \quad (5.7)$$

The coefficient 255 accounts for the fact that the pixel values in \hat{I} are in the range 0 to 255: $\hat{I}(i) \in [0, 255]$. The subscript C indicates image color channel: 1, 2, or 3, corresponding to red, green, blue, respectively. The variable, i , represents a tuple corresponding to pixel location in the C -th channel. Adding the three channels together gives the final damaged image, I .

It would be difficult to enforce effective batch normalization without making accommodations either by modeling and estimating damaged pixel count from layer to layer or turning batch normalization on then off in the encoder stage after a specific number of epochs as in [34]. We found adjusting the damaged pixel values per (5.7) to be an adequate solution.

5.4 Software and Hardware

We trained each unique image network on an NVIDIA Tesla V100 GPU. Each network takes about five hours to train with a training set of 1,000 identical images (the single damaged image cloned 999 times) of size 512 by 512, a batch size of 4, and 40 epochs with 1,000 steps per epoch. The most visually pleasing image result by the 40th epoch is used. The U-Net-like architecture and partial convolution layers are written in Python to leverage Keras [47] operations and network layers with a Tensorflow [50] backend. We used a third-party implementation [51] of the PConvNN with stacked partial convolutional layers and skip

connections as proposed in the NVIDIA paper [34]. This provided us with a working framework, which we then modified to suit our investigation.

The damage and mask elements were created using OpenCV [52]. We expanded upon a suitable mask generator function [53] to create a NumPy array of random shapes and lines. Augmentation and random mask generation are not performed before run-time. Each time a batch of images is loaded, the image data generator applies a unique set of transformations and a random hidden regions mask to each image in the batch.

5.5 Results

Peak signal to noise ratio (PSNR), structural similarity index (SSIM), and other restoration quality metrics are helpful but limited in their ability to indicate true inpainting success. The goal of inpainting is to achieve visually satisfactory results and there are primarily two options: 1) the output image must look realistic but does not have to match the truth, and 2) the output image must be as close to the truth as possible. In other words, given an image of a building with a missing region that encompasses a window, success may be defined as having the true window restored, or success may be defined as having a window that is believable, though not the true window, projected into the region. Nonetheless, we use common inpainting metrics, PSNR and SSIM, to quantitatively compare methods.

To compare our system with a multi-image trained network, we used weights from a third-party implementation [51] of the original PConvNN proposed in [34]. The weights were generated from training the PConvNN on ImageNet [39]. One difference between the original PConvNN and the third-party implementation of PConvNN is that the original network employs He kernel weight initialization [49] whereas the third-party network uses Glorot uniform weight

initialization [54]. Experimentally, we found no significant difference between results from using He initialization and results from using Glorot initialization. Since the third-party implementation is trained on ImageNet using specific parameters, we adjusted our network parameters accordingly for generating PConvNN results.

To compare our system with a non-machine learning method, we used a third-party implementation [55] of PatchMatch based on the original algorithm proposed in [10]. It is important to note that inpainting results generated by the third-party implementations are not exact representations of the original PConvNN and PatchMatch results. However, they are good implementations nonetheless and effectively allow us to provide a relatively fair, albeit imperfect, comparison. Figure 5-3 shows results from performing image restoration on five images using four methods: 1) PM, 2) PConvNN trained on ImNet, which we will shorthand PConv(ImNet), 3) PConvNN trained on the single damaged image using our region hiding technique, partial convolution based padding [48], and local scaling of equation (5.1), which we will shorthand PConv(1) Local, and 4) PConvNN trained on the single damaged image using our region hiding technique, zero padding, and global scaling of equation (5.4), which we will shorthand PConv(1) Global.

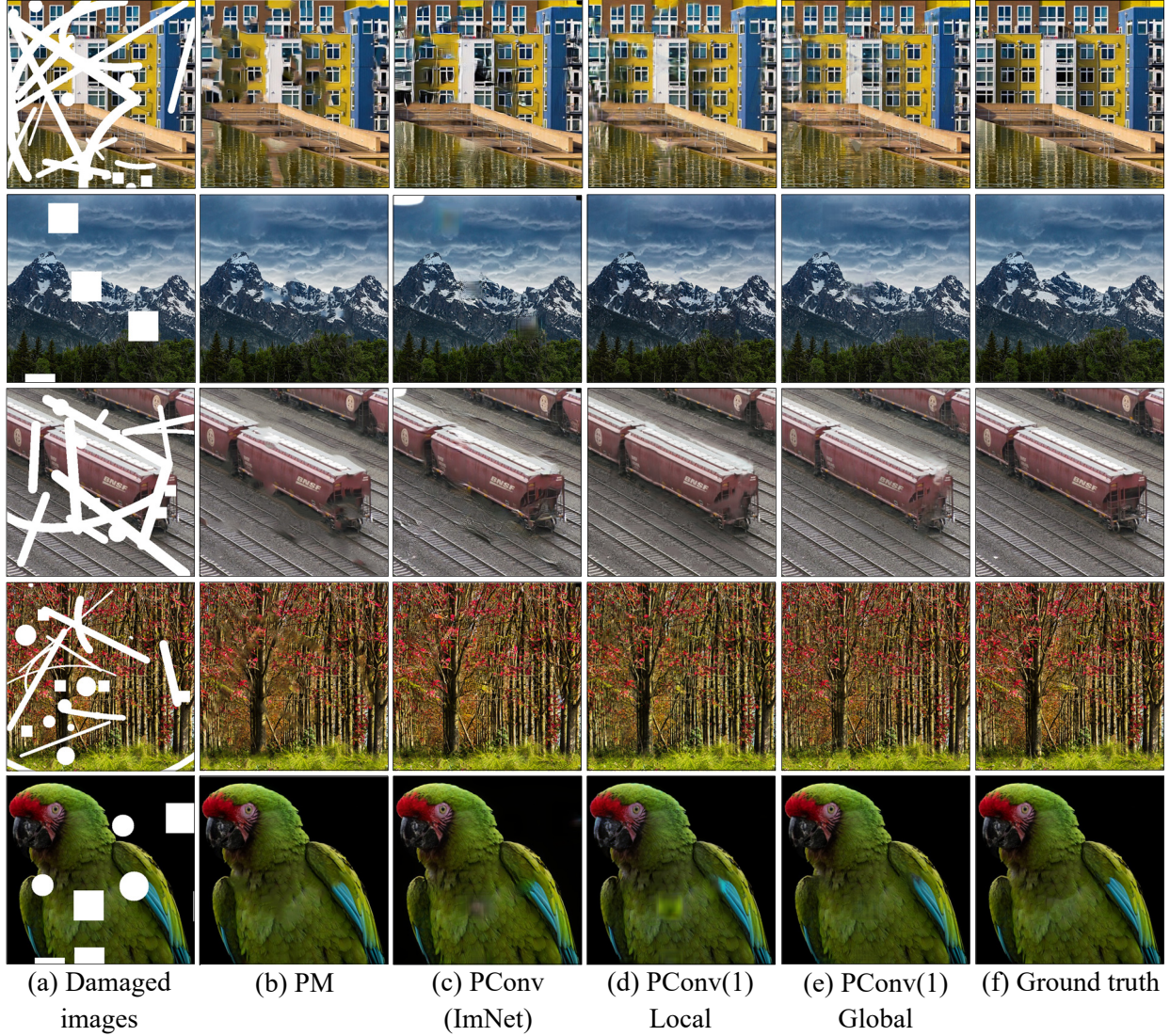


Figure 5-3, Inpainting results for five images⁵. From left to right, (a) damaged images, (b) inpainting results from applying PatchMatch [10], (c) inpainting results from applying multi-image trained PConvNN [34], (d) inpainting results from training PConvNN on the single damaged image using our region hiding technique, partial convolution based padding [48], and local scaling, (e) inpainting results from training PConvNN on the single damaged image using our region hiding technique, zero padding, and global scaling, and (f) ground truth image.

⁵ *Color Block Living* by Sheila Sund MD is licensed under CC BY 2.0. Cropped and resized from original.
Departing Storm by Sheila Sund MD is licensed under CC BY 2.0. Cropped and resized from original.
BNSF Switching Yard by CJ Oliver is licensed under CC BY 2.0. Cropped and resized from original.
Autumn Grove by Sheila Sund MD is licensed under CC BY 2.0. Cropped and resized from original.
Colours by Tomasz Baranowski is licensed under CC BY 2.0. Cropped and resized from original.

5.5.1 Quantitative

We noticed that the inpainting results from the third-party version of PConvNN suffered from black and white patches in the upper corners. In an effort to appropriately and fairly consider the original PConvNN, we compensated for the black and white artifacts in the corners of the ImNet trained PConvNN results by slightly cropping the images to exclude these regions when calculating the PSNR and SSIM for each method's output. Although the values in Table 5-2 were calculated after cropping, the images in all figures of this dissertation are not cropped so that the reader may view entire images.

Table 5-2: Quantitative Inpainting Results

		Damaged image, I	PM	PConv (ImNet)	PConv(1) Local	PConv(1) Global
PSNR	Building	16.76	17.88	17.93	18.78	19.72
	Mountains	23.34	24.11	24.86	22.65	24.44
	Switch Yard	20.66	24.92	23.87	24.48	24.83
	Trees	21.78	21.68	19.90	20.28	20.52
	Bird	27.40	35.01	32.39	30.79	33.83
SSIM	Building	0.74	0.81	0.82	0.83	0.85
	Mountains	0.94	0.95	0.93	0.90	0.93
	Switch Yard	0.80	0.82	0.80	0.82	0.82
	Trees	0.90	0.91	0.88	0.86	0.88
	Bird	0.93	0.99	0.93	0.96	0.98

Table 5-2 shows the PSNR and SSIM scores for each method on the five images in Figure 5-3. It is important to note that the damaged images used to calculate the associated entries in Table 5-2 were created using (5.7), therefore it is possible for the damaged image to seemingly outperform the computational restoration methods quantitatively. However, qualitatively the results favor the computational restoration methods. The PSNR and SSIM

values for each method are comparable despite the varied appearances of their associated images. These differences can be better appreciated in Figure 5-3 and Figure 5-4.

5.5.2 Qualitative

Although PM generally ranks above the other inpainting methods quantitatively, we found that PSNR and SSIM values did not coincide with satisfactory visual results. We have included magnified damaged regions and results of each method for three different images in Figure 5-4. PSNR does not evaluate perceptual quality and therefore predictions that appear disjointed may have a greater PSNR than predictions that appear more natural as can be observed in the second set of images in Figure 5-4. PM scores highest yet the train tracks and train are skewed. PConv(1) Global does a much better job of maintaining structural cohesion.

For some images, there appears to be a trade-off between structural continuity and color consistency within the predicted image. For example, in the first set of images in Figure 5-4, the PConv(1) Global result is structurally cohesive but not as successful at capturing accurate color representations as PM and PConv(ImNet) results. Unfortunately, batch normalization did not prevent this discoloration from occurring. The PConv(1) Local result falls between the two extremes by providing realistic, continuous colors and passable structure. We found that generally, PM and PConv(ImNet) tend to favor the style representation of an input image in their predictions whereas PConv(1) Global tends to favor the content representation of an input image. See our discussion of style and content in section 4.1.

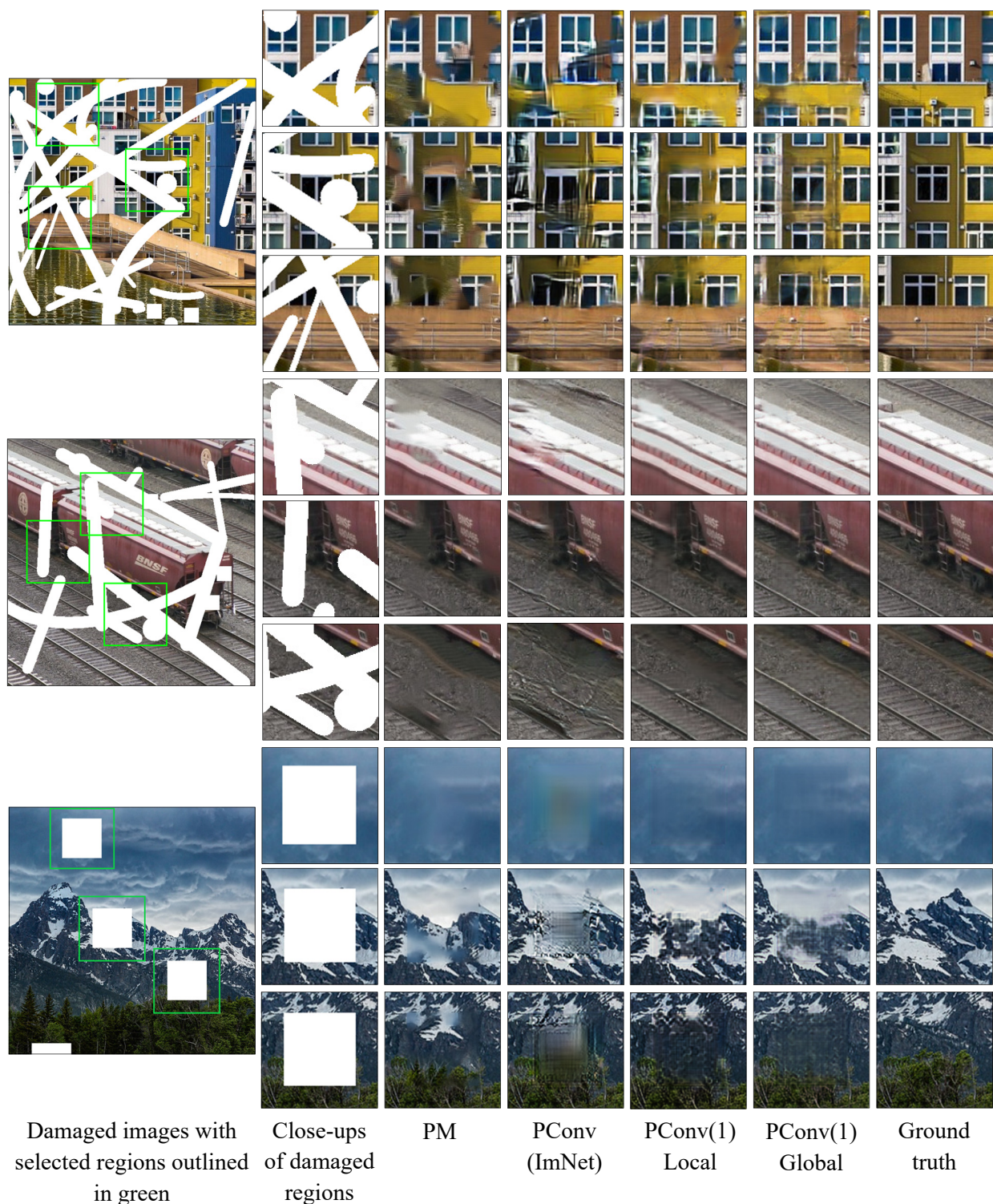


Figure 5-4 Three sets of damaged images with selected regions outlined in green. Close-up images from left to right: selected damaged regions, PM results, PConvNN trained on ImNet results, PConvNN trained on single image using PConv padding and local scaling, PConvNN trained on single image using zero padding and global scaling, and ground truth. Zoom in for better viewing.

5.5.3 Ablation Study

Having little to no style loss generates blurry results as can be observed in column (c) of Figure 5-5. Lack of perceptual loss produces results that are grainy and seem to repeat a grid pattern within the damage, column (b) of Figure 5-5. We did not notice much difference between results with no total variation loss and increased total variation loss although the increased total variation loss occasionally produced results that were less texturally realistic. Generally, utilizing all loss terms produces results that are less blurry, less repetitive, and more realistic.



(a) Close ups of damaged regions (b) No perceptual loss (c) No style loss (d) All loss terms (e) Ground truth

Figure 5-5 Columns from left to right: Close-ups of selected damaged regions, results from training without perceptual loss, results from training without style loss, results from training with all loss terms, corresponding ground truth regions. Zoom in for better viewing.

5.6 Conclusion

Although it may seem excessive to train an entire CNN on a single image to inpaint that image, there are some reasonable motivations to consider. For example, art restoration when the

artist’s work is very stylistic. This situation would require a relatively small data set of images of the artist’s intact paintings as well as an image of the damaged painting. Another potential use for region hiding is object removal, for which only two images are necessary. Furthermore, the benefits of exploring a new concept may launch ideas in other domains.

We provide an example of object removal using our system of region hiding in Figure 5-6. The first image is the testing image which contains both desired and undesired objects [56]. The second image is the training image which was captured at a different time than the first image; perhaps “after the moment had passed.” In this example, we wished to preserve the girl and dog but remove the chair on the right. The second image was captured at a cloudier time and at a slightly different angle than the first image. Therefore, the image containing both desired and undesired elements is brighter than the image containing neither. One way to overcome this is to color normalize the training image based on the testing image. We found that without applying color normalization before training the network and without batch normalization in the encoder stage, the predicted image appeared darker than the testing image. Applying color normalization to the training image did lighten the predicted image but not to the preferred intensity and therefore we further applied color normalization to the output image using the original image for reference.



Figure 5-6 From left to right, top to bottom, original image [56], normalized reference image, original image with unwanted object masked, object removal result from training on the reference image to inpaint the masked image.

Skip connections, which concatenate the weight maps produced in the encoder stage to their counterpart weight maps in the decoder stage, encourage preservation of desired objects in the predicted image. Unfortunately, this also means the presence of shadows from removed objects. Adjustments can be made to overcome artifacts, such as the shadows of removed objects, by fine tuning color matching and batch normalization parameters or simply adjusting the damage mask to encompass problem areas.

In this chapter we have shown that region hiding can generate results that are comparable to other successful inpainting methods without reliance upon large datasets. We have also suggested additional applications for region hiding and provided an example of object removal using our method.

6 Segmentation of Structures in Whole Slide Images of Colon Tissue with U-Net

6.1 Introduction

The National Cancer Institute estimates that colorectal cancer (CRC) will claim the lives of over 51,000 people in the United States in 2019 [57]. According to the American Cancer Society, the risk of developing CRC during one's lifetime is about 1 in 22 for men and 1 in 24 for women. Fortunately, thanks to improved detection methods and treatment options, the number of deaths per year due to CRC is decreasing [58]. Nonetheless, CRC is the second leading cancer cause of death among young adults (ages 20-49 years) and third leading cancer cause of death among all ages in the United States [59] [58]. The 2017 publication of the official *Annual Report to the Nation on the Status of Cancer* concluded that more attention and resources should be given to identifying major risk factors for common cancers such as colorectal, breast, and prostate [60].

Pathology is a subdivision of medical science concerned with the processes and causes of disease. Pathology specialists, or pathologists, examine biopsied tissues, bodily fluids, and organs in an effort to understand and diagnose diseases [61]. Pathologists, therefore, play a crucial role in diagnosing and determining cancer prognosis. Cancer prognosis is an estimated description of how cancer will affect a patient and how the cancer will respond to treatment. The factors that influence prognosis can be separated into two categories: prognostic and predictive.

Prognostic factors include tumor features such as size, location in the body, and grade, as well as patient features such as age and overall health. Predictive factors such as tumor markers, biomarkers, and genetic mutations help pathologists predict how cancer will respond to a certain treatment for a specific patient. For instance, some treatments are only effective for patients with a specific marker or genetic mutation [62] [63] [64]. Thorough multidimensional analysis allows oncologists to assign suitable treatment specifically designed for each unique patient. As expected, timely and accurate predictions are very important.

Currently, OmniPathology and a small team at Chapman University are collaborating to develop an innovative and extremely practical digital system that streamlines colon cancer prognostication. In this dissertation, we focus on one major component of the project; the segmentation of biological structures in whole slide images using image processing and machine learning techniques. We start with an overview and provide important background pertaining to colon cancer and biological structures in context with this portion of the project in 6.2. In sections 6.3 through 6.8, we discuss our current work and results up to this point. Finally, in section 6.9 we describe our goals for this component of the project moving forward and plans of implementation for cell type classification. The overall venture represents a true collaboration of computational, data, and medical sciences. The content of this dissertation is limited to the image segmentation aspect of the project.

6.2 Background and Existing Knowledge

Whole slide imaging is a relatively new technology that allows complete digital scanning of pathology slides to produce whole slide images (WSIs), using state-of-the-art, high definition scanning equipment and software. In April 2017, the Food and Drug Administration (FDA)

approved Philips' IntelliSite Pathology Solution (PIPS) as a whole slide imaging system for primary diagnosis [65]. This gave pathologists in the US the ability to perform primary diagnoses based on images of surgical biopsy slides without the obligation of physically being present to view the tissue under a microscope. Now, pathologists can work remotely, and biopsy slides no longer need to be shipped to specialists or between offices if a second or third opinion is desired. This is big step toward more efficient and accurate diagnosing procedures. Figure 6-1 shows an example of a WSI of colon tissue.

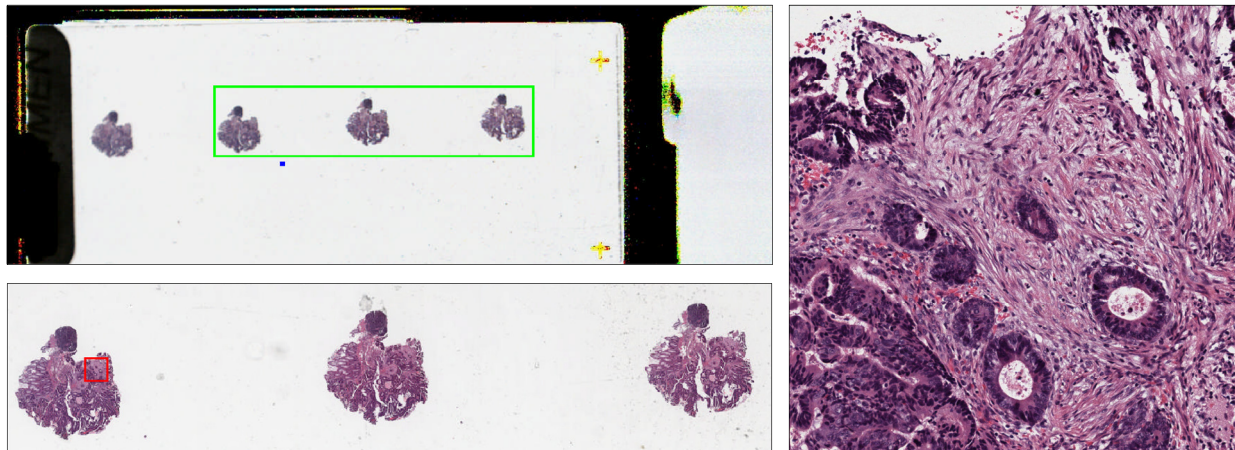


Figure 6-1 Example of WSI of colon tissue sample [66]. Top left: Macro image of the entire slide with three sections of interest indicated with a green rectangle. Bottom left: Thumbnail of three sections from the same tissue sample with a red box around a region of interest within the left-most section. Right: close-up of the region of interest from the actual WSI.

Colon adenocarcinomas are tumors that begin in the glands lining the large intestine. These glands, also known as crypts, are composed of epithelial cells and goblet cells. Goblet cells release mucous out onto the surface of the lumen of the large intestine to lubricate and act as a protective barrier against foreign particles. Figure 6-2 shows a sample of healthy colon tissue. Notice that the epithelium is one-cell thick and appears structurally organized, as if the gland lining is comprised of sheets of cells. This is indicative of healthy tissue. When the

epithelial cells transform into tumors, they no longer maintain the orderly structure of the one-cell thick sheet. This can be observed in the image on the right in Figure 6-3.

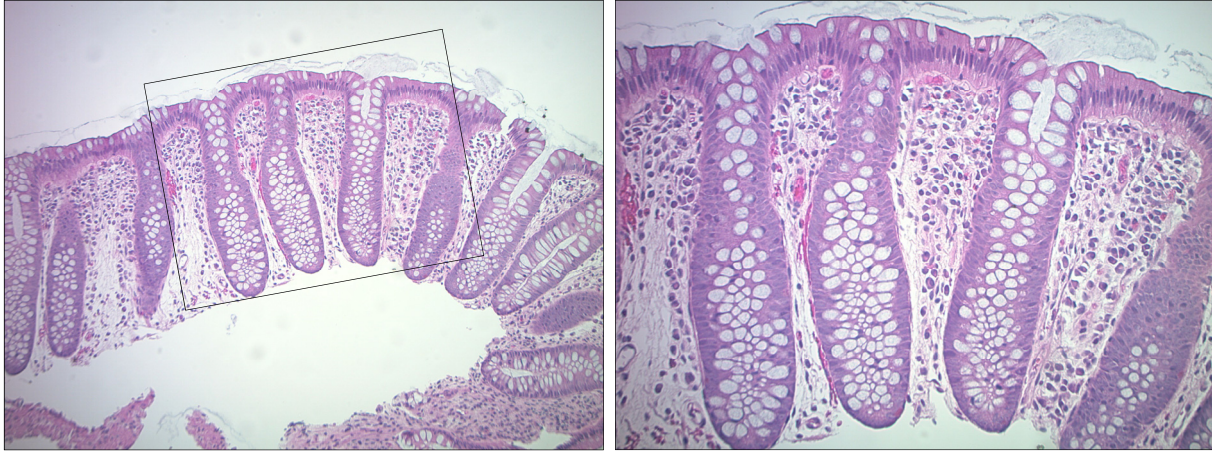


Figure 6-2 WSI of healthy colon tissue from the OmniPathology dataset [67]. Image on left is at 10X magnification and image on right is at 20X magnification. Black rectangle in left image indicates location of the image on the right. Stroma between the glands appears normal with no inflammatory cells. The glands are one-cell thick and appear organized.

In response to the development of tumor cells, the body creates dense, fibrous tissue called desmoplastic stroma (DS) or desmoplastic response (DR), which in and of itself is considered a key indicator of the presence of tumors. Consideration of both tumor and DS and their relationship with each other provides a more complete picture of tumor progression than examining tumor cells alone. Previous research has shown that regarding DS as a predictor of colon cancer progression is prudent. Although researchers do not agree on how DS influences colon cancer invasion, metastasis, recurrence, and progression, they do agree on its importance. DS is thought to be the most important biological factor of malignant neoplasm [68]. While some research articles and studies claim that DS promotes tumor progression by providing nutrients and blood supply to build-up and support the tumor [69] [70] [71], others indicate that DS acts as a barrier protecting healthy tissue from potential cancer diffusion [72]. Perhaps the most accurate interpretation is that DS is responsible for both helping and hindering tumor progression [68].

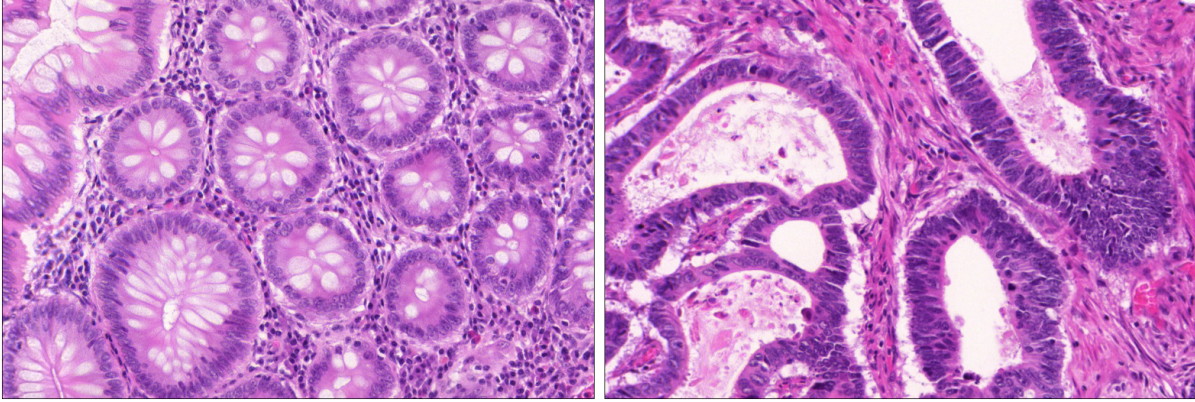


Figure 6-3 Comparison of structures in healthy and unhealthy colon tissues [73]. Left: healthy glands with normal stroma and organized epithelial cells. Right: Malignant glands with stroma containing inflammatory cells and disorganized masses of epithelial cells which have become tumors. When stroma is created in response to cancer, it is called the stromal response, or desmoplasia.

Computational pathology is the use of computational science algorithms and tools to facilitate the objectives of pathology. Currently, the expectations surrounding computational pathology are that it will enable pathologists to receive and report results more quickly, as well as offer new and advanced grading schemes of improved accuracy [74]. Soon, image processing applications will be introduced for extracting information from WSIs that pathologists cannot overtly detect or accurately assess. Noting recent and current developments in computational pathology, we are exploring how our image processing and machine learning algorithms can be utilized within the scope of colon cancer prognostication.

6.3 Segmentation Project Aims

The main structures of interest present in a colon biopsy slide include glands, tumors, and stroma. For this project, our goal is to isolate these structures as well as background artifacts, and further differentiate between cell nuclei within the glands or tumors and cell nuclei within the stroma. Although we do not differentiate between the many specific cell types present in biopsied tissue: smooth muscle cells (SMCs), lymphocytes, fibroblasts, myofibroblasts, plasma

cells, neutrophils, and eosinophils, we do present a useful automatic system for viewing larger structures and cell nuclei independently of each other. In the future, our intention is to create a system that does distinguish between cell types with the added capability of toggling back and forth between segmentation maps from the same WSI to aid in histological analysis. Although outside the scope of this dissertation, potential implementation details of this future goal are discussed in section 6.9.

6.4 Data

The data consists of digital RGB (red, green, blue color space) color images of hematoxylin and eosin (H&E)-stained colon biopsy slides. There are three sources from which we collected the images used in this segmentation project. The segmentation CNN was trained on the Warwick-QU dataset. The Warwick-QU dataset consists of 165 bitmap images of resolution 20X ($0.62005\mu\text{m}/\text{pixel}$), photographed using a Zeiss MIRAX MIDI slide scanner. The images were acquired by a team of pathologists at the University Hospitals Coventry and Warwickshire, UK. The Cancer Digital Slide Archive (CDSA) is a major source of WSIs and associated patient data [66]. The Colon Adenocarcinoma (COAD) dataset consists of 1441 diagnostic H&E-stained WSIs. Each WSI in CDSA is linked to The Cancer Genome Atlas's (TCGA) open access content so that individual patient clinical label data can be extracted for training and validation. Moreover, we have been provided with 30 JPEG image files of various sizes courtesy of Dr. Mohammad Kamal of OmniPathology [67]. Dr. Kamal has also been an invaluable source of information pertaining to cell types and structures within the tissue.

6.5 Approach

We use a CNN to delineate the large structures: glands, stroma, and artifacts, and histogram analysis with thresholding in tandem with the CIE $L^*a^*b^*$ color space [75] to isolate cell nuclei. We rely on pixel intensity to distinguish cells from the rest of the image. The thin strips of biopsied tissue are treated with H&E dyes to color the acidic structures purple and the basic structures pink. Eosin is an acidic dye and is therefore drawn to basic structures in the tissue: most proteins in the cell cytoplasm. Hematoxylin is not basic but it binds to a mordant of aluminum ions which are drawn to acidic structures: DNA in the cell nuclei and RNA in the ribosomes and rough endoplasmic reticulum in the cytoplasm, effectively causing the Hematoxylin to behave as a basic dye [76]. When changing a color image to grayscale, purple manifests in the transformed image as a darker intensity than pink. Because cell nuclei are acidic, they appear dark purple in H&E-stained slides. Therefore, we implement a thresholding algorithm to determine the appropriate grayscale pixel intensity that successfully preserves the dark purple cell nuclei.

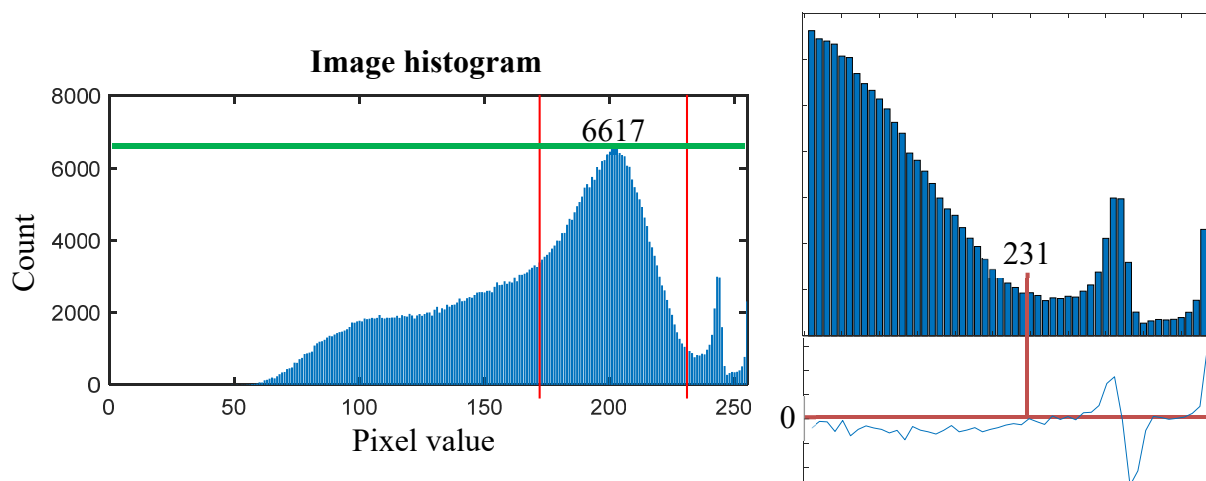


Figure 6-4 An example of a histogram for a region of interest from a WSI. The first threshold is based on the mean pixel value and the second threshold is based on the first zero crossing that occurs after the global maximum. This can be more easily appreciated in the figure on the right which incorporates a plot of the derivative of the histogram below. The slope of the histogram equals zero at pixel value 231.

Color normalization ensures that lab-to-lab staining variability does not impact segmentation results. Reinhard normalization, as described in [77], is a color correction method that applies color features of a reference image to a target image. Both images are transformed from the RGB color space to the $l\alpha\beta$ color space [78], then each element in each channel, l , α , and β , of the target image is shifted by subtracting the mean of all elements in each channel. Next, each element in each channel of the mean-shifted target image is rescaled by the ratio of the target image's standard deviation and the reference image's standard deviation for each channel. Finally, the mean of the reference image for each channel is added to each element in each channel of the target image [77]. The color corrected target image is transformed back to RGB and then to grayscale in preparation for histogram analysis.

After applying Reinhard normalization using a standard H&E-stained slide image for reference, the mean pixel value of the grayscale image, which is used as the first threshold value, is close to 172 for all normalized input images. Pixels that are less than this value are classified as belonging to the glands or cells within the stroma. Experimentally we observed that the pixel value that delineates the background artifact from the rest of the image corresponds to the local minimum following the absolute maximum of the input image histogram. This second threshold value is mathematically determined by differentiating the section of the histogram signal above the absolute maximum with respect to the pixel values. Since the derivative at the location of the local minimum is zero, we assign the pixel value associated with the first zero crossing to the second threshold. When there is no local minimum after the global maximum, a default value of 216 is assigned to the second threshold. This default value was determined experimentally. The segmentation results from our automatic thresholding scheme for the samples in Figure 6-5 are provided in section 6.7.2. The operation is not limited to segmenting structures in images

containing malignant glands. Figure 6-9 in section 6.7.2 shows the histogram and segmentation results for an image containing solely normal colon tissue. The segmentation results for an additional image of normal colon tissue are included in Appendix C.4.

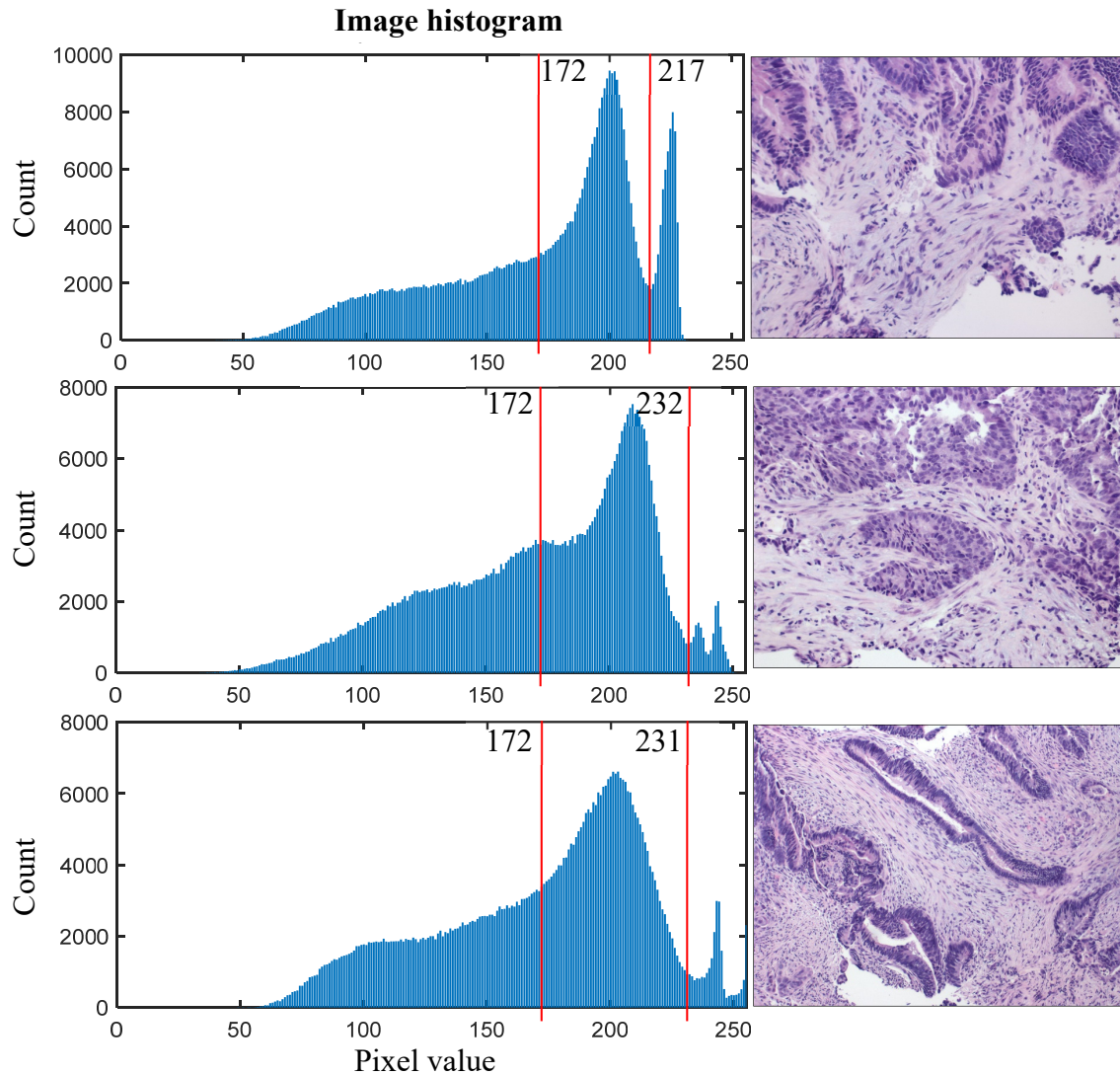


Figure 6-5 The histograms of three normalized images from the OmniPathology dataset [67]. The vertical red bars indicate the pixel values that delineate the three main components of the input images: (1) Glands and cells in the stroma, (2) stroma, and (3) background artifact. The first threshold value is equivalent for each image due to the color normalization operation. The second threshold value corresponds to the local minimum immediately following the absolute maximum of the histogram.

The CIE $L^*a^*b^*$ color space is a three coordinate system like the RGB color space but instead of each channel being descriptive of a specific color; i.e., red, green, and blue, L^* represents luminance in the first layer, a^* , color ranging from green to red in the second layer, and color ranging from blue to yellow in the third layer, b^* . We do not use the CIE $L^*a^*b^*$ color space to delineate structures initially since luminance in the grayscale transformed images is sufficient to distinguish the three main components. Furthermore, at times the background and the stroma are similar in color and would therefore be grouped in the same class. Instead, we utilize the additional luminance thresholding offered by the CIE $L^*a^*b^*$ color space to further isolate cell nuclei from cytoplasm.

6.6 Gland Segmentation U-Net Architecture

The U-Net architecture used for gland segmentation is similar to the architecture utilized in the region hiding inpainting scheme from chapter 5. We again use the encoder-decoder system with stacked convolutional layers and skip connections. However, because we are not trying to fill in missing information, we do not employ the partial convolutional operation in our convolutional layers. The $256 \times 256 \times 3$ color image to be segmented is input into the network at the encoder stage which starts with a convolutional layer that uses zero padding and horizontal and vertical strides of size 2. This process continues until a bottle neck is reached following layer 7, at which point the decoder stage begins. Skip connections concatenate encoder stage output channels to decoder stage input channels after nearest neighbor upsampling.

The last layer of the network connects the network input to the output of the second to last layer, applies the final convolutional layer operation then applies sigmoid activation before outputting the segmentation map result. Sigmoid activation is an appropriate nonlinearity for

binary semantic segmentation tasks because it ensures that output values will be between 0 and 1. Additionally, due to the nature of the sigmoid function curve, individual output values tend to favor one of the two extremes, therefore promoting clear distinction between gland and not gland classification. Table 6-1 shows the architecture of the gland segmentation network that we trained on the GlaS Warwick-QU training dataset [73]. We report quantitative results using the two testing sets from the same source in section 6.7.1.

Table 6-1 Details of Gland Segmentation U-Net Architecture

Layer	Type	Layer output maps (count and size)		Kernel size	Stride	Upsample	BN	Activation
In	Input	3	256×256	--	--	--	--	--
1	Convolution	64	128×128	7×7	2	--	--	ReLU
2	Convolution	128	64×64	5×5	2	--	BN	ReLU
3	Convolution	256	32×32	5×5	2	--	BN	ReLU
4	Convolution	512	16×16	3×3	2	--	BN	ReLU
5	Convolution	512	8×8	3×3	2	--	BN	ReLU
6	Convolution	512	4×4	3×3	2	--	BN	ReLU
7	Convolution	512	2×2	3×3	2	--	BN	ReLU
8	Connect to 6	1024	4×4		--	2	--	--
	Convolution	512	4×4	3×3	1	--	BN	LeakyReLU(0.2)
9	Connect to 5	1024	8×8		--	2	--	--
	Convolution	512	8×8	3×3	1	--	BN	LeakyReLU(0.2)
10	Connect to 4	1024	16×16		--	2	--	--
	Convolution	512	16×16	3×3	1	--	BN	LeakyReLU(0.2)
11	Connect to 3	768	32×32		--	2	--	--
	Convolution	256	32×32	3×3	1	--	BN	LeakyReLU(0.2)
12	Connect to 2	384	64×64		--	2	--	--
	Convolution	128	64×64	3×3	1	--	BN	LeakyReLU(0.2)
13	Connect to 1	192	128×128		--	2	--	--
	Convolution	64	128×128	3×3	1	--	BN	LeakyReLU(0.2)

14	Connect to In	67	256×256		--	2	--	--
Out	Convolution	1	256×256	3×3	1	--	--	Sigmoid

As discussed in chapter 5, the U-Net was initially intended for biomedical image segmentation tasks with sparse datasets [3]. Because we are training on only 85 images, data augmentation is especially important. By applying transformations to the training dataset in order to increase the number of training images, we can encourage the network to learn to segment gland shapes and orientations which may not be represented in the training set. We use Keras’s [47] *ImageDataGenerator* class to apply flips, rotations, and shifts with reflection fill-mode to augment the training dataset.

6.7 Results

In this section we present the results from incorporating the three approaches discussed thus far: (1) Semantic segmentation of glands/not glands in H&E-stained biopsy slides using U-Net with quantitative metrics reported on the GlaS Warwick-QU testing datasets [73], (2) Segmentation of glands, stroma, and background in the images from OmniPathology [67], in regions of interest within WSIs from COAD [66], and in the GlaS Warwick-QU datasets, utilizing our histogram analysis and thresholding algorithm, and (3) Individual cell isolation in regions of interest taken from each of the datasets. Approach (1) is both quantitatively and qualitatively assessed whereas approaches (2) and (3) are qualitatively assessed.

6.7.1 Quantitative Assessment of U-Net Gland Segmentation Results

To quantitatively measure our gland segmentation U-Net performance, we calculate common segmentation metrics: Jaccard similarity coefficient, i.e. intersection over union (IOU), Sørensen-Dice similarity coefficient, boundary F1 score, precision, and recall. Note that although

the Glas Warwick-QU competition site recommends that the F1 score be calculated in terms of true positive (TP), false positive (FP) and false negative (FN) glandular object detection counts, we provide the *boundary* F1 score which is calculated at the pixel level and indicates how closely the boundaries in the predicted segmentation mask match the boundaries in the corresponding ground truth segmentation mask. The boundary F1 score is calculated as follows:

$$BF = \frac{2*precision*recall}{precision+recall}, \quad (6.1)$$

where *precision* is the number of pixels on the boundary of the predicted segmentation mask that are close *enough* to the boundary of the ground truth segmentation mask divided by the predicted boundary length, and *recall* is the number of pixels on the boundary of the ground truth segmentation mask that are close *enough* to the boundary of the predicted segmentation mask divided by the ground truth boundary length. In other words, *precision* is the number of true positives detected divided by the total number of positives detected, both true and false:

$$precision = \frac{TP}{TP+FP}, \quad (6.2)$$

and *recall* is the number of true positives detected divided by the total number of positives, both detected and undetected:

$$recall = \frac{TP}{TP+FN}. \quad (6.3)$$

The pixel level metrics were calculated for each individual image then the resulting values were averaged over the specific testing set and listed in Table 6-2.

Table 6-2 Metric values for test sets A and B of GlaS Warwick-QU

	Jaccard similarity coefficient (IOU)	Dice similarity coefficient	Boundary F1 score	Precision	Recall
Test set A	0.8195	0.8955	0.7737	0.7309	0.8360
Test set B	0.8168	0.8899	0.7415	0.7092	0.7849

To demonstrate the difference between pixel level BF score and object level F1 score values, we have included three samples showing our predicted segmentation masks atop the ground truth segmentation masks in Figure 6-6. The overlaid segmentation masks were generated in MATLAB with *imshowpair* function [18]. In image (a), the predicted segmentation received a perfect score for object level F1 since the model successfully detected six glands in the slide image. However, although the contours of the ground truth and the segmentation prediction are closely aligned, they are not perfectly matched, therefore the BF score is 0.91. Image (b) shows a case in which the object level F1 score would be less than 1 since the small gland touching the right border, middle from the top, is less than 50% overlapped by the predicted segmentation mask and is therefore considered a false negative. The two glands in the right upper corner would nonetheless be considered true positives since each is overlapped by at least 50% of the predicted mask. Image (c) shows the case in which there are five true positives detected and an extraneous region that the network incorrectly identified as a gland. Again, this would result in both a lower BF score and a lower F1 score.



Figure 6-6 Three samples to show the difference between predicted segmentation results and ground truth segmentation. White indicates overlap of the two segmentation masks (true positives), purple indicates regions that the ground truth mask considers glandular tissue but that the prediction mask does not (false negatives), and green indicates regions that the prediction mask considers glandular tissue but that the ground truth mask does not (false positives). We have also included corresponding pixel level metrics and object level metrics in Table 6-3.

Table 6-3 Quantitative comparison of three segmentation mask overlap samples

	Image (a)	Image (b)	Image (c)
IOU	0.9356	0.9197	0.7031
Dice	0.9667	0.9582	0.8257
BF	0.9116	0.8596	0.7026
Precision	0.9214	0.8902	0.5463
Recall	0.9021	0.8310	0.9840
Object IOU	1.0	0.8333	0.8333
Object Dice	1.0	0.9091	0.9091
Object F1	1.0	0.9091	0.9091
Object Precision	1.0	1.0	0.8333
Object Recall	1.0	0.8333	1.0

Whereas the boundary F1 score quantitatively expresses how close the predicted gland boundaries are to the actual gland boundaries, the Jaccard similarity coefficient and Sørensen-Dice similarity coefficient measure the overlap of the two segmentation masks. The Jaccard similarity coefficient is calculated as the number of pixels in the prediction mask that overlap with the pixels in the ground truth mask divided by the number of pixels in their union:

$$jaccard(GT, MASK) = \frac{|GT \cap MASK|}{|GT \cup MASK|} = \frac{TP}{TP + FP + FN}. \quad (6.4)$$

In our case, the pixel level Jaccard similarity coefficient can be visually appreciated as the area of the white region divided by the combined areas of the white, purple, and green regions shown in Figure 6-6. The Dice similarity coefficient is like Jaccard similarity but double counts the overlapping region in both the numerator and denominator:

$$dice(GT, MASK) = \frac{2|GT \cap MASK|}{|GT| + |MASK|} = \frac{2*TP}{2*TP + FP + FN}. \quad (6.5)$$

Object level Dice similarity is equivalent to object level F1 score.

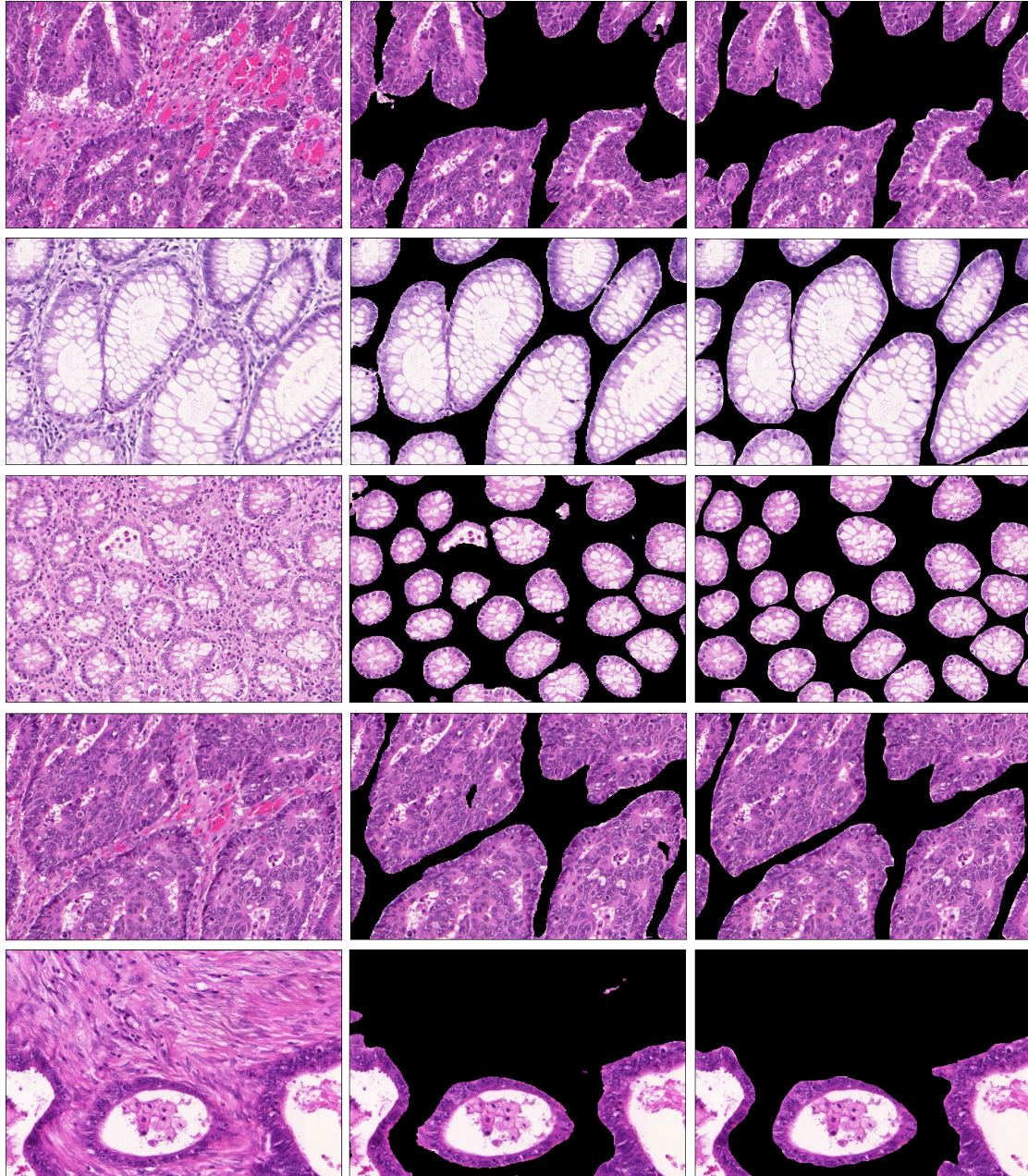


Figure 6-7 Successful segmentation results for five images taken from test set A of [73]. The left column shows the input images, the center column shows segmentation prediction mask atop the input image, and the right column shows the ground truth segmentation mask atop the input image.

Examination of metrics and what they represent about the results is important to our overall mission. It may be the case that perfect alignment of contours between the prediction and the ground truth is unnecessary or perhaps it is indeed necessary but there is some room for error.

To successfully detect and isolate cell nuclei post gland segmentation, the ability of the network to perform well at the pixel level is key. False positives may sound innocuous compared to false negatives but depending on the locations of the cells we are trying to detect, i.e., the glands or the stroma, closely aligning with the true boundaries of the gland objects is important for cell isolation in both locations.

6.7.2 Qualitative Assessment of Cell Isolation Results

We do not have annotated data to quantitatively validate our background and cell nuclei segmentation results. However, we have included several samples of our results and have based our qualitative assessment on feedback from Dr. Kamal over several meetings and conversations. We will first provide results from our automatic segmentation system via histogram analysis and thresholding as described in section 6.5. Then, we will further isolate cell nuclei by transforming the region of interest to the CIE $L^*a^*b^*$ color space and removing the light blue pixels from our detection mask. Finally, we incorporate the U-Net segmentation results so that we may differentiate between cells in the glands/tumors and cells in the stroma.

The results from applying our histogram thresholding segmentation algorithm, described in section 6.5, to the three images in Figure 6-5 are shown in Figure 6-8. The algorithm successfully distinguishes between the three main components. Figure 6-10 applies both histogram thresholding and our pretrained CNN to a 3000×3000 pixel region of interest (ROI) from a WSI downloaded from the CDSA [66]. The first step is to apply Reinhard color normalization to the ROI then input the image into the automatic thresholding program. The output consists of three images focusing on the three components: the first is the normalized input with all but the stroma masked, the second is the normalized input with all but the tumors

and cells masked, and the third is the normalized input with all but the background artifact masked. At this point, we are ready to combine the masks from both segmentation methods.

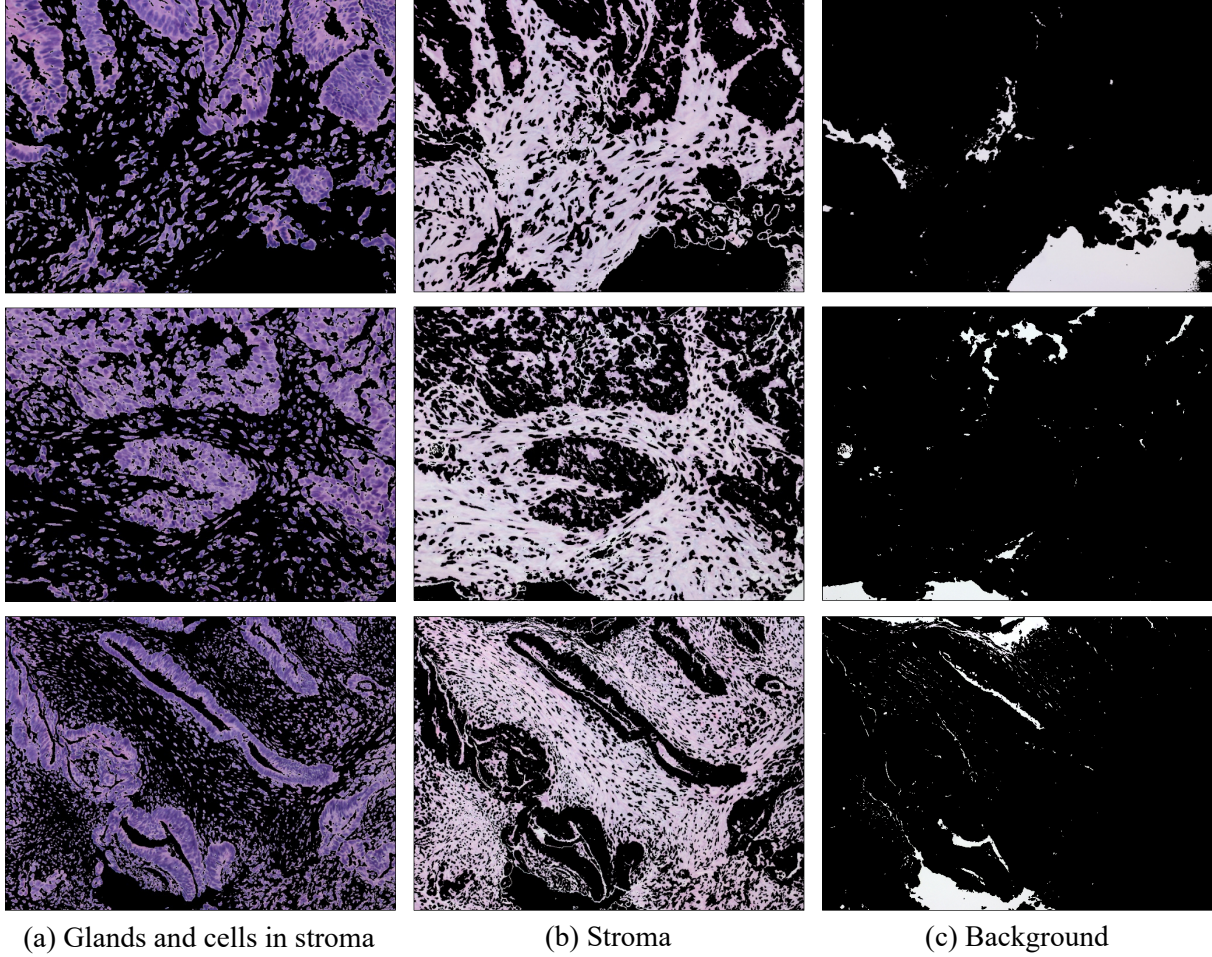


Figure 6-8 Segmentation results from using our histogram thresholding algorithm on the three images in Figure 6-5. The top two images are at 20X magnification while the bottom image is at 10X magnification. All three images are from the OmniPathology dataset [67].

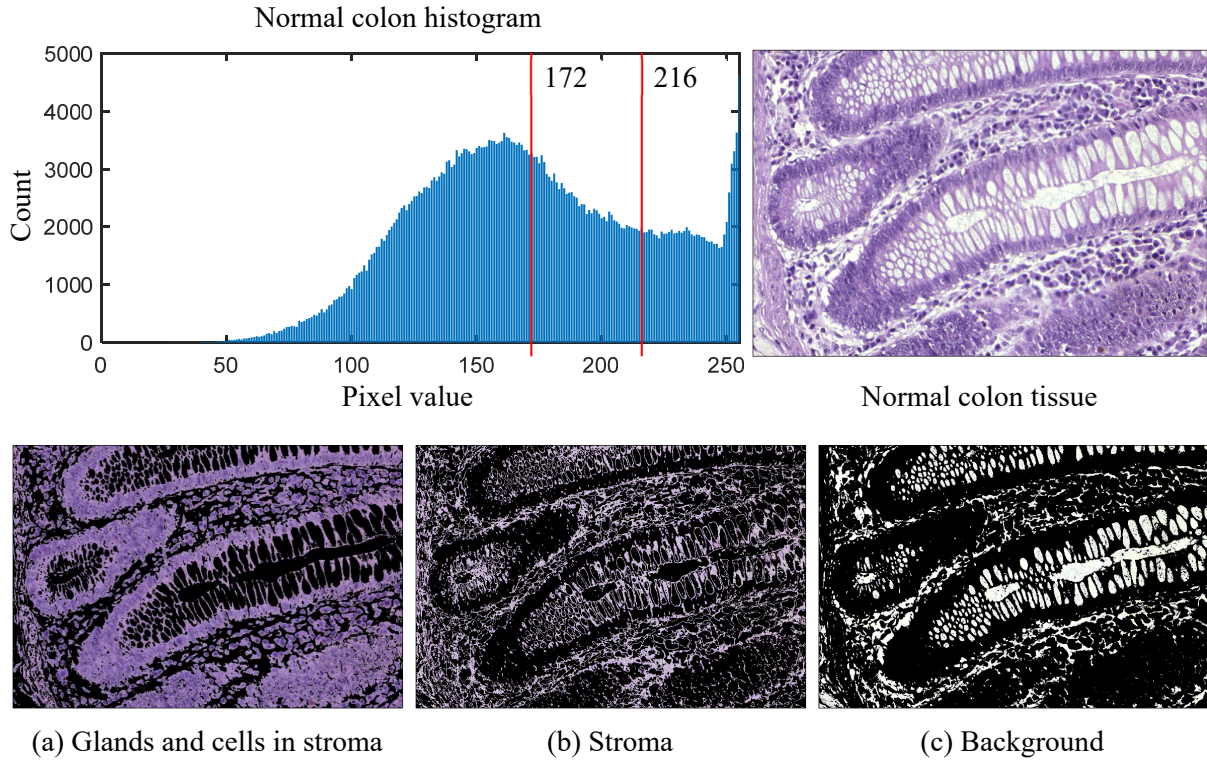


Figure 6-9 Histogram of normalized image of benign tissue from test set B of [73] and segmentation results from utilizing our automatic thresholding algorithm. The vertical red bars indicate the pixel values that delineate the three main components: (a) Glands and cells in stroma, (b) stroma, and (c) background. Since there is no local minimum after the global maximum, which occurs at pixel value 255, a default value of 216 is assigned to the second threshold.

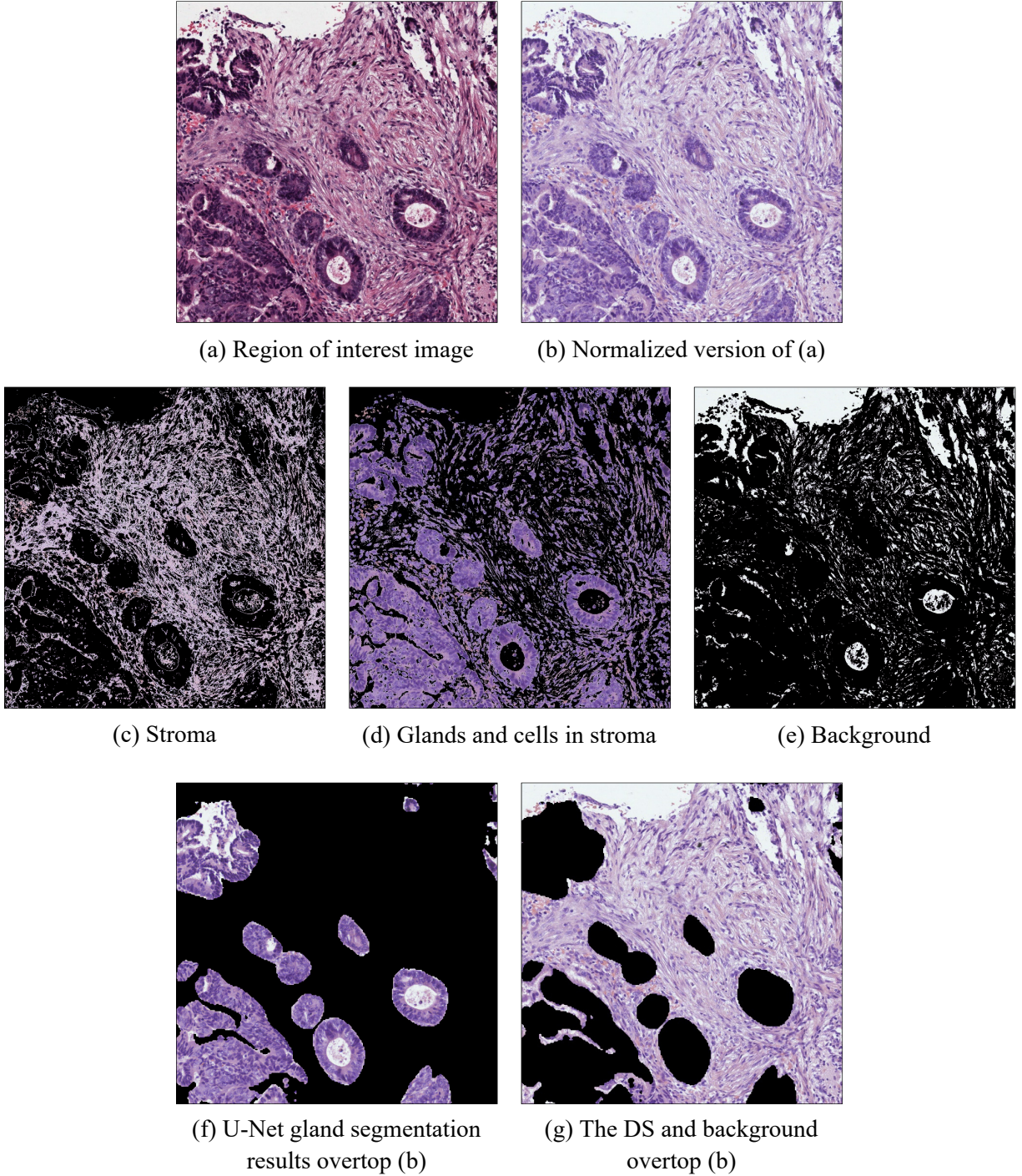


Figure 6-10 Example of utilizing the two segmentation methods together to isolate various structures in a region of interest taken from a WSI from the CDSA [66]. From here, we can distinguish between cells in the stroma and cells in the tumors.

We will now examine the cell nuclei isolation method which relies on the CIE $L^*a^*b^*$ color space. As mentioned in section 6.5, we use CIE $L^*a^*b^*$ because it represents pixels in a

way that measures color and luminance separately. Figure 6-11 shows the progression of cell nuclei isolation by location. Cell type can be difficult to discern without the context of the surrounding stroma. However, an artificial neural network can be trained to classify cell types based on characteristics that humans do not intuitively recognize as descriptive; shape, for example. Figure 6-12 shows a close-up from the bottom right image in Figure 6-11. We follow-up with the notion of utilizing shape to classify cell types in section 6.9.

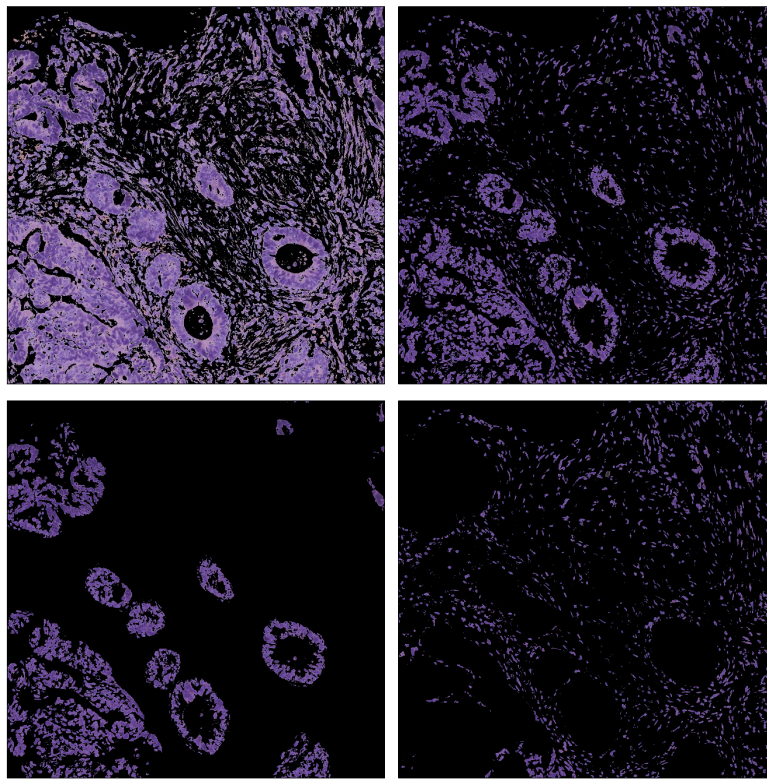


Figure 6-11 Cell nuclei isolation. Top left to bottom right: Input image, results from filtering relatively *light* pixels from the CIE $L^*a^*b^*$ transformed input image, application of the mask in part (f) of Figure 6-10 to show isolated cell nuclei located in the tumors, and application of the mask in part (g) to show isolated cell nuclei located in the stroma.

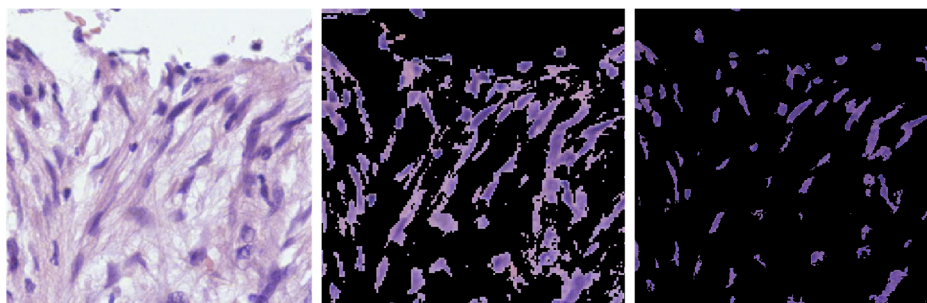


Figure 6-12 Close-up of cells in the stroma. The rightmost image is the result from isolating cell nuclei in the center image.

In some cases, isolating the cell nuclei may not be as desirable as viewing the cells with their cytoplasm. Depending on the application, it may be more appropriate to elect for less severe cell segmentation. Figure 6-13 shows an example of this. The center image is a close-up of the result from our histogram thresholding program with the glands/tumors masked. The image on the right shows the cell nuclei segmentation results. The center image does a better job at capturing both cell nuclei and cytoplasm. However, it does not separate individual cells as well as the image on the right. Nonetheless, the cell nuclei isolation method has masked many cells that the histogram analysis approach successfully detected. Looking closely, the image on the right has removed smooth muscle cells and many fibroblasts, which the center image preserves. Rather than display more of our results in the text of this dissertation, we have provided additional samples from the three approaches in Appendix C.

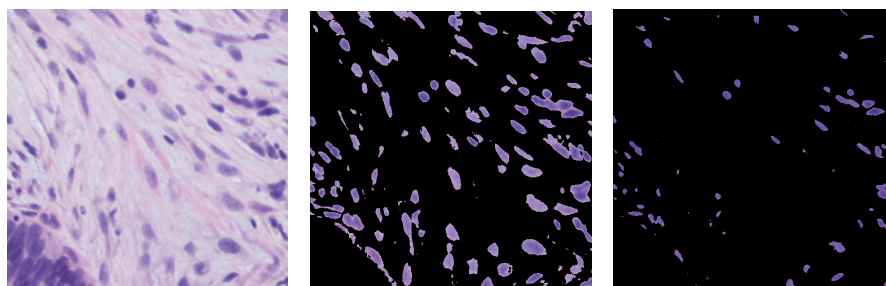


Figure 6-13 Close-up of cells in the stroma taken from the segmentation results shown in Appendix C. The rightmost image is the result from isolating cell nuclei in the center image. Depending on the application, the center image may be more desirable than the image on the right.

6.8 Software and Hardware

To select, view, and save regions of interest, we use OpenSlide [79] with an intuitive MATLAB wrapper [80]. We trained the segmentation U-Net on Schmid’s HPC (high performance computing) cluster with an NVIDIA Tesla V100 GPU. The network takes about 40 minutes to train with a training set of 85 color images of various sizes that are resized to 256 by 256, with a batch size of 5, 20 epochs, 1000 steps per epoch, and simple ℓ_1 loss. We experimented with binary cross entropy and a combination of ℓ_1 and IOU losses but simple ℓ_1 loss worked best. We utilize Adam optimizer with a constant learning rate of 0.0002. Batch normalization is employed in both the encoder and decoder stages. Otherwise, we did not apply any preprocessing normalization to the datasets. The U-Net was written in Python to use Keras [47] operations and network layers with a Tensorflow backend [50]. To segment an image in the testing phase, it takes 2 seconds per 256×256 color image (2.37 seconds/iteration) and 2 seconds per 512×512 color image (2.69 seconds/iteration).


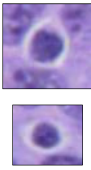
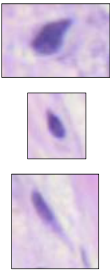
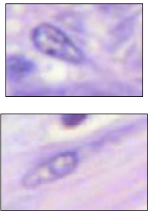
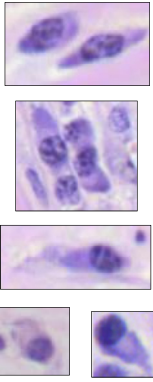

The Genomic Data Commons (GDC) data transfer tool [81] was used to transfer WSIs and associated patient clinical data from the CDSA COAD repository [66]. WSIs are in SVS format and were extracted from Aperio using OpenSlide 3.4.1 [79]. Many of the individual SVS files contain up to four images associated with the slide: the WSI, label, macro image, and thumbnail. Figure 6-1 shows the macro image (top left), the thumbnail (bottom left), and a 3000×3000 pixels patch from the WSI (right). We implement small cellular structures segmentation in MATLAB [18]. Before applying our histogram thresholding technique, we first normalized all images using Reinhard normalization in HistomicsTK [82].

6.9 Cell Type Classification

Cell detection can be accomplished by locating cell nuclei based on pixel intensity.

Classifying cells into specific cell types is a more complicated task because cell type cannot be determined by pixel intensity alone. The traits that separate one cell type from another in an image pertain to color, shape, and size. For example, fibroblasts are visually distinct because of their tear-drop wispy appearance. Depending on the lab where the staining occurs, the technician doing the staining, or even the unique variations among fibroblasts themselves, they may range from quite light purple to very dark purple. Fibroblasts also vary in size which means classification must rely heavily upon shape. Table 6-4 shows examples of cell types that are common in WSIs.

Table 6-4 Images of cell types

SMCs	Lymphocytes	Fibroblasts	Myofibroblasts	Plasma Cells	Neutrophils and Eosinophils
 <p>Long cigar shaped nucleus.</p>	 <p>Large nucleus, very little cytoplasm.</p>	 <p>Tear drop shaped, can appear wispy.</p>	 <p>Similar to fibroblasts with less hematoxylin dye in the nucleus.</p>	 <p>Clock face nucleus, lots of cytoplasm.</p>	 <p>Eosinophils look like neutrophils but with more eosin in nucleus, hence the name.</p>

We now present a prospective algorithm which can be used to classify cells by type.

Pathologists select regions of interest from WSIs to create a suitable dataset. The regions of

interest are subjected to color normalization, histogram analysis, and preliminary segmentation before being separated into sub-images and sent to pathologists for annotation, Figure 6-14.

- 1) Remove the stroma and slide background from the image so we are just left with cell nuclei.
- 2) Apply segmentation masks to distinguish between cells located in the glands/tumors and cells located in the stroma.
- 3) Use blob detection to pinpoint the location of each unique cell and store these locations in a matrix.
- 4) Classify each cell by its type using a CNN.

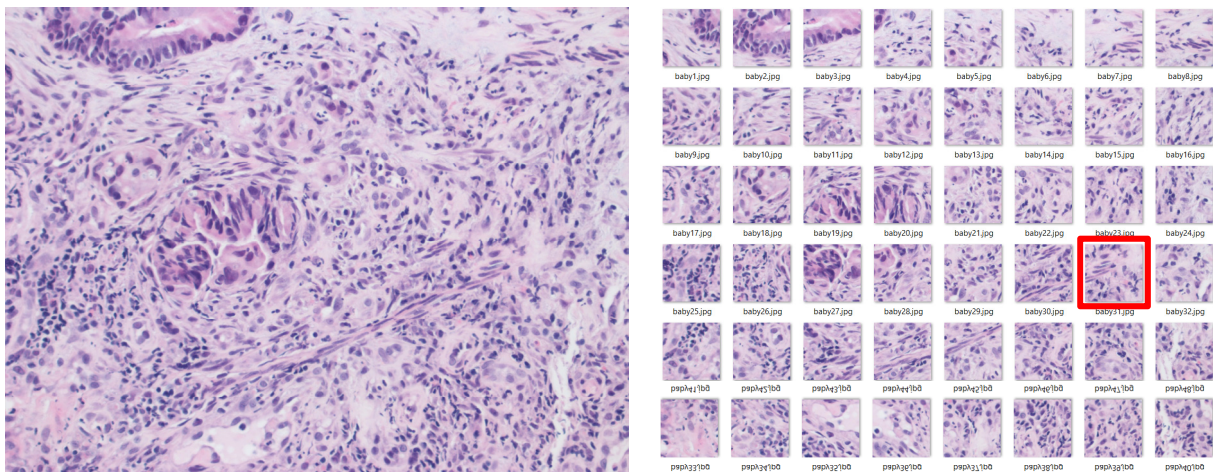


Figure 6-14 Regions of interest are subjected to color normalization, histogram analysis, and preliminary segmentation before being separated into sub-images and sent to pathologists for annotation.

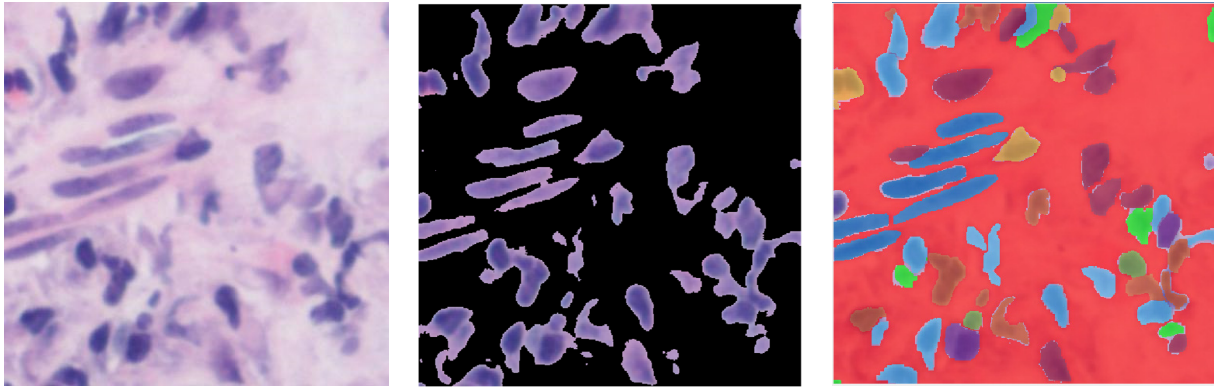


Figure 6-15 From left to right: Sub-image from region of interest outlined in red in Figure 6-14, automatic thresholding results, and *simulated* semantic segmentation of tissue by cell type. (Important: The image on the right is not an actual semantic segmentation result.)

Once all cells are classified, the user can choose to display “blobs” (or cells) that fit a particular classification. This would also be useful for cell counting by type, Figure 6-16.

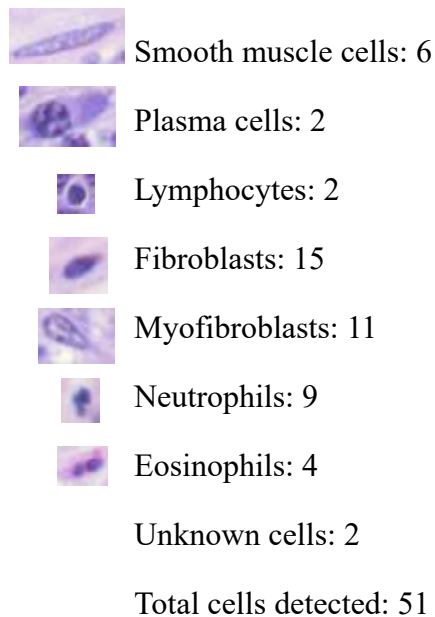


Figure 6-16 Simulated cell counting results.

Although we can use computational science tools and algorithms to make the process easier, time and effort must be invested into the examination and annotation of each sub-image by pathologists

7 Conclusion

In this dissertation we covered several image restoration techniques and leveraged previous methods to design and implement our own systems. We began with a survey of traditional, non-machine learning approaches to image restoration, and focused on algorithms that could be applied specifically to missing and damaged regions inpainting. We demonstrated techniques both within their proposed frameworks and beyond, through application within our inpainting systems. We proposed and implemented an automatic damaged regions detection algorithm by building upon and improving previous methods. We adjusted the SVD cosine similarity matrix to incorporate side by side and stacked pixel neighborhoods so that both horizontal and vertical damage is detected, instead of focusing solely on vertical damage. Furthermore, we applied the algorithm outside of its intended use, detecting cracks in images of walls and pavement, by including scanned damaged photographs and digital images of finely cracked paintings.

We developed a unique, simple, and effective solution to inpainting damaged regions at the borders of scanned photographs and cracked paintings by applying PDE boundary conditions outward from the interior of the repaired image counterintuitively. For images containing splotches and dots, we recognized the usefulness of the inverted Laplacian magnitude matrix, used in image compression and decoding, and proposed applying it to automatic damaged regions detection to preserve edge locations. Although our system does not completely overcome manual inpainting mask creation, it offers a time-saving and precise solution to fully manual

mask creation, especially for difficult wide-spread damage, by automatically generating approximate inpainting masks that can be easily modified.

The main contribution of this dissertation is an image inpainting technique that relies on machine learning. After a short introduction to GANs and encoder-decoders, we discussed common loss functions used for training CNNs in image-to-image translation tasks. We introduce the U-Net and defend our decision to utilize its encoder-decoder architecture with stacked partial convolutional layers and skip connections. We describe an original approach to image inpainting using a novel tactic called region hiding. Region hiding encourages the network to learn to inpaint the damaged image by forcing it to make predictions about regions that it cannot see, but that are indeed intact. The network is awarded for correctly predicting the hidden regions based on non-damaged unhidden regions. The success of the region hiding method with the PConvNN can be observed in the results provided.

The last written component of this dissertation embodies image processing and machine learning methods seemingly unrelated to inpainting. However, both image restoration and image segmentation are examples of translation where underlying information from the input image must be purposefully represented in the output image. Yet, it is also necessary that specific parts of the signal be filtered or removed from the input image altogether, be it noise, damage, texture, or color. We began this final component by seriously considering the prevalence of colon cancer and its increasing occurrence in young adults. We provided applicable colon histology terminology and helpful background information to facilitate reader understanding before discussing goals and implementation details. To segment large structures in regions of interest taken from WSIs, we again employed the U-Net with convolutional layers. Working with a

sparse training dataset compelled us to utilize augmentation techniques. Our segmentation success can be both quantitatively and qualitatively appreciated in the results provided.

Through experimentation, we observed that the histograms associated with WSIs tended to conform to a similar pattern. To delineate color intensities, which also distinguish tissue properties due to the nature of H&E staining, we calculated two thresholds. This divided the histogram into three parts corresponding to three specific structures: glands (and/or tumors) and cells in the stroma, stroma, and slide background. After converting the input image to the CIE $L^*a^*b^*$ color space, we further isolated individual cell nuclei by operating on the luminosity layer of the masked gland image to filter out lighter pixels. Using these cell isolation results together with the glands segmentation results allowed us to distinguish between cells in the glands and cells in the stroma. We conclude our WSI structures segmentation component by revealing future goals for this research, including cell type classification.

References

- [1] C. Martin-King and M. Allali, "Automatic damaged region detection and inpainting," in *The 20th International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV'16)*, Las Vegas, 2016.
- [2] C. Martin-King and M. Allali, "Region hiding for image inpainting via single-image training of U-Net," in *The 2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, 2019.
- [3] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 9351, pp. 234-241, 2015.
- [4] P. Kainz, M. Pfeiffer and M. Urschler, "Segmentation and classification of colon glands with deep convolutional neural networks and total variation regularization," *PeerJ*, 2017.
- [5] M. Bertalmio, L. Vese and G. Sapiro, "Simultaneous structure and texture image inpainting," *IEEE Transactions On Image Processing*, vol. 12, no. 8, pp. 882-889, 2003.
- [6] L. A. Vese and S. J. Osher, "Modeling textures with total variation minimization and oscillating patterns in image processing," *Journal of Scientific Computing*, vol. 19, pp. 553-572, 2003.
- [7] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *IEEE International Conference on Computer Vision*, Corfu, 1999.
- [8] A. Gersho, Artist, *Barbara*. [Art].
- [9] E. Karaca and M. A. Tunga, "An interpolation-based texture and pattern preserving algorithm for inpainting color images," *Expert Systems With Applications*, vol. 91, pp. 223-234, January 2018.
- [10] C. Barnes, E. Shechtman, A. Finkelstein and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," in *ACM Transactions on Graphics (Proc. SIGGRAPH)*, New Orleans, 2009.
- [11] L. I. Rudin, S. Osher and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259-268, 1992.

- [12] T. F. Chan, J. Shen and H.-M. Zhou, "Total variation wavelet inpainting," *Journal of Mathematical Imaging and Vision*, vol. 25, no. 1, pp. 107-125, 2006.
- [13] M. Mainberger, A. Bruhn, J. Weickert and S. Forchhammer, "Edge-based compression of cartoon-like images with homogeneous diffusion," in *Fachrichtung 6.1 -- Mathematik*, Saarbrücken, Universität des Saarlandes, 2010.
- [14] L. Hoeltgen, M. Mainberger, S. Hoffmann, J. Weickert, C. Hoo Tang, S. Setzer, D. Johannsen, F. Neumann and B. Doerr, "Optimising spatial and tonal data for PDE-based inpainting," in *Lecture Notes in Computer Science*, vol. 6667, Springer, Berlin, Heidelberg, 2012, pp. 26-37.
- [15] *Trui*. [Art].
- [16] R. W. Floyd and L. S. Steinberg, "An adaptive algorithm for spatial grayscale," in *Proceedings of SID*, Seattle, 1976.
- [17] M. G. Padalkar, M. A. Zaveri and M. V. Joshi, "SVD based automatic detection of target regions for image inpainting," in *Asian Conference on Computer Vision; Computer vision - ACCV 2012 Workshops*, Daejeon, 2012.
- [18] MATLAB, *version 9.5.0.1033004 (R2018b) Update 2*, Natick: MathWorks Inc., 2018.
- [19] M. Rutledge, Artist, *A Crack in the Pavement of a Chequered Neighborhood*. [Art]. Flickr.com, 2008.
- [20] alien_sunset, Artist, *Pavement cracks I (2)*. [Art]. Flickr.com, 2012.
- [21] Unknown, Artist, *Unknown Oil Painting of Eye*. [Art].
- [22] J. Butler, Artist, *Torn Victorian Photo Pre-restoration*. [Art]. PhotoValet.
- [23] simpleinsomnia, Artist, *Photo booth portrait of a chubby man 2, "Dad 1940"*. [Art]. Flickr.com, 2016.
- [24] Unknown, Artist, *Boy in Overalls*. [Art].
- [25] Unknown, Artist, *GIRLBADI*. [Art].
- [26] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, 2014.

- [27] S. Iizuka, E. Simo-Serra and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.
- [28] H. Li, G. L. L. L. and Y. Yu, "Context-aware semantic inpainting," *IEEE Transactions on Cybernetics*, vol. 49, no. 12, pp. 4398-4411, 2018.
- [29] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu and T. S. Huang, "Generative image inpainting with contextual attention," *arXiv preprint arXiv:1801.07892*, 2018.
- [30] C. Yang, Y. Song, X. Liu, Q. Tang and C.-C. J. Kuo, "Image inpainting using block-wise procedural training with annealed adversarial counterpart," *arXiv preprint*, March 2018.
- [31] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes and J. Luo, "Foreground-aware image inpainting," *ArXiv*, 22 April 2019.
- [32] X.-J. Mao, C. Shen and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *29th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, 2016.
- [33] Y. Liu, J. Pan and Z. Su, "Deep blind image inpainting," *CoRR*, vol. abs/1712.09078v1, 25 December 2017.
- [34] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," *ECCV 2018*, 2018.
- [35] P. Isola, J.-Y. Zhu, A. A. Efros and T. Zhou, "Image-to-image translation with conditional adversarial networks," *arXiv:1611.07004v3*, 2018.
- [36] L. A. Gatys, A. S. Ecker and M. Bethge, "A neural algorithm of artistic style," *Journal of Vision*, 2015.
- [37] J. Johnson, A. Alahi and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, Amsterdam, 2016.
- [38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, San Diego, 2015.
- [39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211-252, 2015.
- [40] S. Sund, Artist, *Color Block Living*. [Art]. Flickr, 2015.

- [41] S. Sund, Artist, *Departing Storm*. [Art]. Flickr.com, 2018.
- [42] C. J. Oliver, Artist, *BNSF Switching Yard*. [Art]. Flickr.com, 2010.
- [43] S. Sund, Artist, *Autumn Grove*. [Art]. Flickr.com, 2015.
- [44] T. Baranowski, Artist, *Colours*. [Art]. Flickr.com, 2018.
- [45] D. Ulyanov, A. Vedaldi and V. Lempitsky, "Deep image prior," *arXiv:1711.10925v3*, 2018.
- [46] A. W. Harley, K. G. Derpanis and I. Kokkinos, "Segmentation-aware convolutional networks using local attention masks," in *IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.
- [47] F. Chollet, *Keras*, GitHub, 2015.
- [48] G. Liu, K. J. Shih, T.-C. Wang, F. A. Reda, K. Sapra, Z. Yu, A. Tao and B. Catanzaro, "Partial convolution based padding," NVIDIA Corp., Santa Clara, 2018.
- [49] K. He, X. Zhang, S. Ren and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.
- [50] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, V. Fernanda, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.
- [51] M. Gruber, "PConv-Keras," 2018. [Online]. Available: <https://github.com/MathiasGruber/PConv-Keras>. [Accessed 7 January 2019].
- [52] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [53] M. Gruber, "PConv-Keras," 2018. [Online]. Available: <https://github.com/MathiasGruber/PConv-Keras/blob/master/libs/util.py>. [Accessed 07 January 2019].
- [54] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy, 2010.

- [55] younesse-cv, "PatchMatch," 9 September 2013. [Online]. Available: <https://github.com/younesse-cv/PatchMatch>. [Accessed June 2019].
- [56] C. Martin-King, Artist, *Emma and Yakimo*. [Art]. 2019.
- [57] SEER, "Cancer Stat Facts: Colorectal Cancer," National Cancer Institute, 2019. [Online]. Available: <https://seer.cancer.gov/statfacts/html/colorect.html>. [Accessed 11 October 2019].
- [58] American Cancer Society Medical and Editorial Content Team, "Key Statistics for Colorectal Cancer," American Cancer Society, 21 February 2018. [Online]. Available: <https://www.cancer.org/cancer/colon-rectal-cancer/about/key-statistics.html>. [Accessed 27 April 2018].
- [59] E. M. Ward, R. L. Sherman, S. J. Henley, A. Jemal, D. A. Siegel, E. J. Feuer, A. U. Firth, B. A. Kohler, S. Scott, J. Ma, R. N. Anderson, V. Benard and K. A. Cronin, "Annual Report to the Nation on the Status of Cancer, Featuring Cancer in Men and Women Age 20–49 Years," *JNCI: Journal of the National Cancer Institute*, 2019.
- [60] A. Jemal, E. M. Ward, C. J. Johnson, K. A. Cronin, J. Ma, A. B. Ryerson, A. Mariotto, A. J. Lake, R. Wilson, R. L. Sherman, R. N. Anderson, S. J. Henley, B. A. Kohler, L. Penberthy, E. J. Feuer and H. K. Weir, "Annual Report to the Nation on the Status of Cancer, 1975–2014, Featuring Survival," *JNCI: Journal of the National Cancer Institute*, vol. 109, no. 9, 2017.
- [61] McGill Department of Pathology, "What is Pathology?," McGill University, 2018. [Online]. Available: <https://www.mcgill.ca/pathology/about/definition>. [Accessed 27 April 2018].
- [62] National Cancer Institute, "Understanding Cancer Prognosis," National Cancer Institute, 24 November 2014. [Online]. Available: <https://www.cancer.gov/about-cancer/diagnosis-staging/prognosis>. [Accessed 27 April 2018].
- [63] L. Simms, H. Barraclough and G. Ramaswamy, "Biostatistics Primer: What a Clinician Ought to Know—Prognostic and Predictive Factors," *Journal of Thoracic Oncology*, vol. 8, no. 6, pp. 808-813, 2013.
- [64] E. Nalejska, E. Mączyńska and M. A. Lewandowska, "Prognostic and Predictive Biomarkers: Tools in Personalized Oncology," *Molecular Diagnostics and Therapy*, vol. 18, no. 3, pp. 273-284, 2014.
- [65] V. N. Newitt, "Whole slide imaging for primary diagnosis: 'Now it is happening'," CAPS Today, College of American Pathologists, May 2017. [Online]. Available: <http://www.captodayonline.com/whole-slide-imaging-primary-diagnosis-now-happening/>. [Accessed 21 April 2018].

- [66] Emory Winship Cancer Institute, "Cancer Digital Slide Archive," Emory Winship Cancer Institute, [Online]. Available: <http://cancer.digitalslidearchive.net/>. [Accessed 2 May 2018].
- [67] M. Kamal, *OmniPathology dataset*, Pasadena: OmniPathology, 2017.
- [68] M. Zippi, G. De Toma, G. Minervini, C. Cassieri, R. Pica, D. Colarusso, S. Stock and P. Crispino, "Desmoplasia Influenced Recurrence of Disease and Mortality in Stage III Colorectal Cancer within Five Years after Surgery and Adjuvant Therapy," *The Saudi Journal of Gastroenterology*, vol. 23, no. 1, pp. 39-44, 2017.
- [69] J. Conti and G. Thomas, "The Role of Tumour Stroma in Colorectal Cancer Invasion and Metastasis," *Cancers*, vol. 3, no. 2, pp. 2160-2168, 2011.
- [70] P. P. Provenzano, K. W. Eliceiri, J. M. Campbell, D. R. Inman, J. G. White and P. J. Keely, "Collagen Reorganization at the Tumor-Stromal Interface Facilitates Local Invasion," *BMC Medicine*, vol. 4, no. 38, 2006.
- [71] C. Rupp, M. Scherzer, A. Rudisch, C. Unger, C. Haslinger, M. Schweifer, M. Artaker, H. Nivarthi, R. Moriggl, M. Hengstschlager, D. Kerjaschki, W. Sommergruber, H. Dolznig and P. Garin-Chesa, "IGFBP7, a novel tumor stroma marker, with growth-promoting effects in colon cancer through a paracrine tumor–stroma interaction," *Oncogene*, vol. 34, pp. 815-825, 2015.
- [72] A. Caporale, A. Ciardi, A. Vestri, M. Ruperto and A. Giuliani, "Quantitative Investigation of Desmoplasia as a Prognostic Indicator in Colorectal Cancer," *Journal of Investigative Surgery*, vol. 23, pp. 105-109, 2010.
- [73] "GlaS Warwick-QU Dataset," Department of Computer Science, University of Warwick, 2015. [Online]. Available: <https://warwick.ac.uk/fac/sci/dcs/research/tia/glascontest/download/>. [Accessed 2019].
- [74] J. van Laak , N. Rajpoot and D. Vossen, "The Promise Of Computational Pathology: Part 1," *The Pathologist*, no. 118, January 2018.
- [75] International Commission on Illumination, *CIE L*a*b**, Vienna, 1976.
- [76] Faculty of Biological Sciences, University of Leeds, "The Histology Guide," University of Leeds, 2003. [Online]. Available: https://www.histology.leeds.ac.uk/what-is-histology/H_and_E.php. [Accessed 2019].
- [77] E. Reinhard, M. Ashikhmin, B. Gooch and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34-41, 2001.

- [78] D. L. Ruderman, T. W. Cronin and C.-C. Chiao, "Statistics of cone responses to natural images: implications for visual coding," *Journal of the optical Society of America*, vol. 15, no. 8, pp. 2036-2045, 1998.
- [79] A. Goode, B. Gilbert, J. Harkes, D. Jukic and M. Satyanarayanan, "OpenSlide: A vendor-neutral software foundation for digital pathology," *Journal of Pathology Informatics*, vol. 4, no. 1, p. 27, 2013.
- [80] D. Forsberg, "openslide-matlab," github, 2016. [Online]. Available: <https://github.com/fordanic/openslide-matlab>. [Accessed 2019].
- [81] National Cancer Institute: Genomic Data Commons, *GDC Data Transfer Tool*, National Cancer Institute.
- [82] B. Helba, D. Gutman, D. Manthey, D. R. Chittajallu, J. Beezeley, L. Cooper, S. Lee, Z. Mullen and M. Amgad, "HistomicsTK," Digital Slide Archive, [Online]. Available: <https://github.com/DigitalSlideArchive/HistomicsTK>. [Accessed 2018].
- [83] J.-S. Ou, W.-S. Chen, B. Pan and Y.-G. Li, "A new image inpainting algorithm based on DCT similar patches features," in *2016 12th International Conference on Computational Intelligence and Security (CIS)*, Wuxi, 2016.
- [84] F. Guichard, L. Moisan and J.-M. Morel, "A Review of P.D.E. Models in Image Processing and Image Analysis," *Journal de Physique IV (Proceedings)*, vol. 12, no. 1, pp. 137-154, 2002.
- [85] K. Sirinukunwattana, D. Snead and N. Rajpoot, "A Stochastic Polygons Model for Glandular Structures in Colon Histology Images," *IEEE Transactions on Medical Imaging*, 2015.
- [86] C.-B. Schönlieb, "Applying modern PDE techniques to digital image restoration," MathWorks, 2012. [Online]. Available: <https://www.mathworks.com/company/newsletters/articles/applying-modern-pde-techniques-to-digital-image-restoration.html>.
- [87] F. Berntsson and G. Baravdish, "Coefficient identification in PDEs applied to image inpainting," *Applied Mathematics and Computation*, vol. 242, pp. 227-235, 2014.
- [88] J. Alexander, Artist, *Fruit*. [Art]. Flickr, 2013.
- [89] K. Sirinukunwattana, J. P. W. Pluim, H. Chen, X. Qi, P. Heng, Y. Guo, L. Wang, B. J. Matuszewski, E. Bruni, U. Sanchez, A. Böhm, O. Ronneberger, B. B. Cheikh, D. Racoceanu, P. Kainz, M. Pfeiffer, M. Urschler, D. R. J. Snead and N. Rajpoot, "Gland Segmentation in Colon Histology Images: The GlaS Challenge Contest," *Medical Image Analysis*, vol. 35, pp. 489-502, 2017.

- [90] F. Bijl, Artist, *Ingredients*. [Art]. Flickr, 2007.
- [91] V. K. Alilou and F. Yaghmaee, "Non-texture image inpainting using histogram of oriented gradients," *Journal of Visual Communication and Image Representation*, vol. 48, pp. 43-53, October 2017.
- [92] A. Althouse, Artist, *Yellow Flower*. [Art]. Flickr.com, 2007.
- [93] P. W. Wong, "Image quantization, halftoning, and printing," in *Handbook of Image & Video Processing*, San Diego, Academic Press, 2000, pp. 657-667.

Appendix A. Inpainting with the Discrete Heat Transfer Equation

In Appendix A we derive the discretized boundary condition equations from section 2.1 as well as the discrete diffusion equations used to inpaint the pixels within the boundary. As mentioned in section 2.1, the heat transfer equation is defined as $u_t = u_{xx} + u_{yy}$. We now switch our focus to image processing and introduce the Laplacian of a two-dimensional function (or image), $f(x, y)$:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \quad (\text{A.1})$$

Equation (A.1) is similar to the heat transfer equation but is not time-dependent. In fact, Laplace's equation, $\nabla^2 f = 0$, describes the steady state of the heat transfer equation, where $t = t_0 = 0$. This is the initial condition, or the initial temperature at each location. In image processing, the initial condition is the function $f(x, y, t_0)$ which is equivalent to the input image before diffusion-based inpainting is applied.

A.1 Preliminary Equations and Definitions

To aid in the derivation of equations (2.1) through (2.4), we first present some useful definitions. The derivative of a continuous, single-variable function $f(x)$, is:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}. \quad (\text{A.2})$$

Similarly, the partial derivative with respect to x of a continuous, two-variable function $f(x, y)$, is:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h, y) - f(x, y)}{h}. \quad (\text{A.3})$$

The Taylor series expansion of a function f about a point x is,

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f^{(3)}(x) + \dots \quad (\text{A.4})$$

Suppose that $f = f(x)$ is a twice continuously differentiable function. According to Taylor's theorem, we can approximate the function with the second order Taylor series approximation,

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x). \quad (\text{A.5})$$

Solving for $f'(x)$, we get the forward difference approximation,

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x). \quad (\text{A.6})$$

If we replace h with $-h$ in equation (A.5) and again solve for $f'(x)$, we get the backward difference approximation,

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{h}{2} f''(x). \quad (\text{A.7})$$

Setting the forward and backward difference approximations equal to each other and solving for $f''(x)$ we have,

$$\frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) = \frac{f(x) - f(x-h)}{h} + \frac{h}{2} f''(x), \quad (\text{A.8})$$

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \quad (\text{A.9})$$

Similarly, for two variables, we utilize the forward difference approximation to solve for f_x :

$$f(x+h, y) = f(x, y) + hf_x(x, y) + \frac{h^2}{2}f_{xx}(x, y), \quad (\text{A.10})$$

$$\frac{f(x+h, y) - f(x, y)}{h} = f_x(x, y) + \frac{h}{2}f_{xx}(x, y), \quad (\text{A.11})$$

$$f_x(x, y) = \frac{f(x+h, y) - f(x, y)}{h} - \frac{h}{2}f_{xx}(x, y). \quad (\text{A.12})$$

Likewise, we utilize the backward difference approximation to solve for f_x :

$$f(x-h, y) = f(x, y) - hf_x(x, y) + \frac{h^2}{2}f_{xx}(x, y), \quad (\text{A.13})$$

$$\frac{f(x, y) - f(x-h, y)}{h} = f_x(x, y) - \frac{h}{2}f_{xx}(x, y), \quad (\text{A.14})$$

$$f_x(x, y) = \frac{f(x, y) - f(x-h, y)}{h} + \frac{h}{2}f_{xx}(x, y), \quad (\text{A.15})$$

Setting the first partial derivatives of the forward and backward difference approximations equal to each other, we have:

$$\frac{f(x+h, y) - f(x, y)}{h} - \frac{h}{2}f_{xx}(x, y) = \frac{f(x, y) - f(x-h, y)}{h} + \frac{h}{2}f_{xx}(x, y), \quad (\text{A.16})$$

$$\frac{f(x+h, y) - 2f(x, y) + f(x-h, y)}{h} = hf_{xx}(x, y), \quad (\text{A.17})$$

$$\frac{f(x+h, y) - 2f(x, y) + f(x-h, y)}{h^2} = f_{xx}(x, y), \quad (\text{A.18})$$

$$f_{xx}(x, y) = \frac{f(x+h, y) - 2f(x, y) + f(x-h, y)}{h^2}. \quad (\text{A.19})$$

Now, we introduce a third variable, t , which is the time component relevant to the heat-transfer process and diffusion-based image inpainting. Consider the function $f(x, y, t)$, with

second-order Taylor series approximations. The forward and backward difference approximations, respectively, are:

$$f(x + h, y, t) = f(x, y, t) + hf_x(x, y, t) + \frac{h^2}{2} f_{xx}(x, y, t), \quad (\text{A.20})$$

$$f(x - h, y, t) = f(x, y, t) - hf_x(x, y, t) + \frac{h^2}{2} f_{xx}(x, y, t). \quad (\text{A.21})$$

Although the time variable t is in the third position of the function's input tuple, which is associated with channel indices, we limit our derivation to two dimensional images and therefore recognize t as descriptive of varying time rather than switching through color channels.

Building from equation (A.19), we have the time dependent equation:

$$f_{xx}(x, y, t) = \frac{f(x+h,y,t)-2f(x,y,t)+f(x-h,y,t)}{h^2}, \quad (\text{A.22})$$

where $h > 0$ and tiny.

Now, we end up with the nice collection:

$$f_{xx}(x, y, t) = \frac{f(x+h,y,t)-2f(x,y,t)+f(x-h,y,t)}{h^2}, \quad (\text{A.23})$$

$$f_{yy}(x, y, t) = \frac{f(x,y+k,t)-2f(x,y,t)+f(x,y-k,t)}{k^2}, \quad (\text{A.24})$$

$$f_t(x, y, t) = \frac{f(x,y,t+r)-f(x,y,t)}{r}. \quad (\text{A.25})$$

Typically, $h, k, r > 0$, are very small. However, in image processing, the spatial variables, h and k , must be discrete values since images cannot be represented as continuous signals. The distance from one pixel to the next is one. Thus, the practical version of our collection is:

$$f_{xx}(x, y, t) = f(x + 1, y, t) - 2f(x, y, t) + f(x - 1, y, t), \quad (\text{A.26})$$

$$f_{yy}(x, y, t) = f(x, y + 1, t) - 2f(x, y, t) + f(x, y - 1, t), \quad (\text{A.27})$$

with $f_t(x, y, t)$ remaining as defined in equation (A.25).

A.2 Boundary Conditions

We will now examine the boundary conditions for the example in Figure 2-1. The boundary conditions must be heeded before the interior of the missing region is inpainted. Once the one-pixel thick border is inpainted accordingly, it is not subjected to further reassignment over iterations as the interior of the boundary is. Therefore, since the time variable has no effect on the value of f at the boundaries, we denote $f(x, y, t)$ as

$$f(x, y) = \frac{1}{4}(f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1)). \quad (\text{A.28})$$

This indicates spatial location information and we will maintain that $t \geq 0$. Suppose that the damage is confined to an $m \times n$ rectangular region where $x = i$ is the top boundary of the damage, $x = i + m$ is the bottom boundary, $y = j$ is the left boundary, and $y = j + n$ is the right boundary. The boundary conditions are:

$$f(i, y) = f(i + m, y) = f(x, j) = f(x, j + n) = 0, \quad (\text{A.29})$$

where $i \leq x \leq i + m, j \leq y \leq j + n$.

Missing or damaged pixels should not be used to interpolate other missing or damaged pixels. So, to heed the boundary conditions and maintain edges in the surrounding regions, we shift the first-derivative forward and backward difference equations and set them equal to each other. For the top boundary, $x = i$, we have:

$$f_x(i, y) = f(i + 1, y) - f(i, y), \quad (\text{A.30})$$

$$f_x(i, y) = f(i, y) - f(i - 1, y), \quad (\text{A.31})$$

$$f(i + 1, y) - f(i, y) = f(i, y) - f(i - 1, y), \quad (\text{A.32})$$

$$0 = f(i + 1, y) - 2f(i, y) + f(i - 1, y). \quad (\text{A.33})$$

Shift and solve for $f(i, y)$:

$$0 = f(i, y) - 2f(i - 1, y) + f(i - 2, y), \quad (\text{A.34})$$

$$f(i, y) = 2f(i - 1, y) - f(i - 2, y). \quad (\text{A.35})$$

Recall that this is specifically for the top boundary going downward into the missing region (assuming the pixel values located at $f(i - 1, y)$ and $f(i - 2, y)$ are known for $j \leq y \leq j + n$). To feed information from the exterior of the bottom boundary into the missing boundary above it, $x = i + m$, we again find the forward and backward difference equations then shift.

$$f_x(i + m, y) = f(i + m + 1, y) - f(i + m, y), \quad (\text{A.36})$$

$$f_x(i + m, y) = f(i + m, y) - f(i + m - 1, y), \quad (\text{A.37})$$

$$f(i + m + 1, y) - f(i + m, y) = f(i + m, y) - f(i + m - 1, y), \quad (\text{A.38})$$

$$0 = f(i + m + 1, y) - 2f(i + m, y) + f(i + m - 1, y). \quad (\text{A.39})$$

Shift and solve for $f(i + m, y)$:

$$0 = f(i + m + 2, y) - 2f(i + m + 1, y) + f(i + m, y), \quad (\text{A.40})$$

$$f(i + m, y) = 2f(i + m + 1, y) - f(i + m + 2, y), \quad (\text{A.41})$$

for $j \leq y \leq j + n$. Although it seems counterintuitive, it is important to remember that for the bottom boundary, the function must be shifted in the opposite direction of the top boundary so that known pixel values ($f(i + m + 1, y)$ and $f(i + m + 2, y)$) are utilized. To feed information from the left into the missing region to the right, we adopt the same process as was done for the top and bottom boundaries and arrive at the equation:

$$f(x, j) = 2f(x, j - 1) - f(x, j - 2), \quad (\text{A.42})$$

where $i \leq x \leq i + m$. To feed information from the left into the missing region to the right, the equation is,

$$f(x, j + n) = 2f(x, j + n + 1) - f(x, j + n + 2), \quad (\text{A.43})$$

where $i \leq x \leq i + m$. Ideally, the boundary conditions ensure edge-continuity between the regions of known and unknown pixel values as the missing information is filled in.

A.3 Inpainting the Interior Region

To inpaint the interior of the boundary, we now attain the discrete representation of the heat transfer equation, $f_t = f_{xx} + f_{yy}$, by substituting (A.25) through (A.27) for f_t , f_{xx} , and f_{yy} , respectively, then solving for $f(x, y, t + r)$:

$$\begin{aligned} \frac{f(x, y, t+r) - f(x, y, t)}{r} &= f(x + 1, y, t) - 2f(x, y, t) + f(x - 1, y, t) + \\ &\quad f(x, y + 1, t) - 2f(x, y, t) + f(x, y - 1, t) \\ &= f(x + 1, y, t) + f(x - 1, y, t) + f(x, y + 1, t) + \\ &\quad f(x, y - 1, t) - 4f(x, y, t), \end{aligned} \quad (\text{A.44})$$

$$f(x, y, t + r) = f(x, y, t) + r(f(x + 1, y, t) + f(x - 1, y, t) + f(x, y + 1, t) + f(x, y - 1, t) - 4f(x, y, t)), \quad (\text{A.45})$$

where r is a small number between 0 and 1, such that a large number of steps (theoretically approaching infinity, although not in practice) in tandem with the small step size, r , can be used to gradually fill in the missing information in an iterative process descriptive of heat transfer over time. Since the boundaries have already been repaired, they do not need to be included in the inpainting algorithm. Therefore, $i + 1 \leq x \leq i + m - 1$ and $j + 1 \leq y \leq y + n - 1$.

Appendix B. Total Variation

B.1 Total Variation Flow

As discussed in section 2.3, total variation regularization is utilized to suppress noise in an image while preserving edges. In Appendix B we analyze the total variation flow equation and provide a short discussion with examples to reveal its influence at the pixel level. The total variation flow equation is

$$f_t = \operatorname{div} \left(\frac{\nabla f}{|\nabla f|} \right). \quad (\text{B.1})$$

Here we provide relevant definitions to aid in forthcoming derivations. The divergence of a two-dimensional function f is defined as:

$$\begin{aligned} \operatorname{div}(f) &= \nabla \cdot f \\ &= \left\langle \frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right\rangle \cdot f \\ &= f_x + f_y. \end{aligned} \quad (\text{B.2})$$

The gradient of f is the vector:

$$\nabla f = \langle f_x, f_y \rangle, \quad (\text{B.3})$$

and the norm (Euclidean) of the gradient of f is:

$$|\nabla f| = \sqrt{f_x^2 + f_y^2}. \quad (\text{B.4})$$

With these definitions in mind, let us proceed with our derivation of f_t . Following from equation (B.1) we have:

$$\begin{aligned} f_t &= \text{div} \left(\frac{1}{\sqrt{f_x^2 + f_y^2}} \langle f_x, f_y \rangle \right) \\ &= \frac{\partial}{\partial x} \left(\frac{f_x}{\sqrt{f_x^2 + f_y^2}} \right) + \frac{\partial}{\partial y} \left(\frac{f_y}{\sqrt{f_x^2 + f_y^2}} \right) \end{aligned} \quad (\text{B.5})$$

Component-wise expansion of the partial derivatives gives us:

$$\frac{\partial}{\partial x} \left(\frac{f_x}{\sqrt{f_x^2 + f_y^2}} \right) = \frac{f_{xx}\sqrt{f_x^2 + f_y^2} - f_x^2 f_{xx}(f_x^2 + f_y^2)^{-\frac{1}{2}} - f_x f_y f_{xy}(f_x^2 + f_y^2)^{-\frac{1}{2}}}{f_x^2 + f_y^2}, \quad (\text{B.6})$$

and

$$\frac{\partial}{\partial y} \left(\frac{f_y}{\sqrt{f_x^2 + f_y^2}} \right) = \frac{f_{yy}\sqrt{f_x^2 + f_y^2} - f_y^2 f_{yy}(f_x^2 + f_y^2)^{-\frac{1}{2}} - f_x f_y f_{xy}(f_x^2 + f_y^2)^{-\frac{1}{2}}}{f_x^2 + f_y^2}. \quad (\text{B.7})$$

Substituting back into equation (B.5) we have:

$$\begin{aligned} f_t &= \frac{f_{xx}\sqrt{f_x^2 + f_y^2} - f_x^2(f_x^2 + f_y^2)^{-\frac{1}{2}}f_{xx} - f_x f_y f_{xy}(f_x^2 + f_y^2)^{-\frac{1}{2}}}{f_x^2 + f_y^2} + \\ &\quad \frac{f_{yy}\sqrt{f_x^2 + f_y^2} - f_y^2(f_x^2 + f_y^2)^{-\frac{1}{2}}f_{yy} - f_x f_y f_{xy}(f_x^2 + f_y^2)^{-\frac{1}{2}}}{f_x^2 + f_y^2} \\ &= \frac{1}{f_x^2 + f_y^2} \left(f_{xx}\sqrt{f_x^2 + f_y^2} - f_x^2(f_x^2 + f_y^2)^{-\frac{1}{2}}f_{xx} + f_{yy}\sqrt{f_x^2 + f_y^2} - \right. \\ &\quad \left. f_y^2(f_x^2 + f_y^2)^{-\frac{1}{2}}f_{yy} - 2f_x f_y f_{xy}(f_x^2 + f_y^2)^{-\frac{1}{2}} \right) \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{(f_x^2 + f_y^2)^{\frac{3}{2}}} (f_{xx}(f_x^2 + f_y^2) - f_x^2 f_{xx} + f_{yy}(f_x^2 + f_y^2) - f_y^2 f_{yy} - \\
&\quad 2f_x f_y f_{xy}) \\
&= \frac{1}{(f_x^2 + f_y^2)^{\frac{3}{2}}} (f_{xx}(f_x^2 + f_y^2 - f_x^2) + f_{yy}(f_x^2 + f_y^2 - f_y^2) - 2f_x f_y f_{xy}) \\
&= \frac{1}{(f_x^2 + f_y^2)^{\frac{3}{2}}} (f_{xx}f_y^2 + f_{yy}f_x^2 - 2f_x f_y f_{xy}) \tag{B.8}
\end{aligned}$$

Now that we have arrived at this expanded representation of f_t , we will transition to the discrete version by incorporating equations (A.26) and (A.27) as well as the following new equations:

$$f_x(x, y, t) = \frac{1}{2} (f(x + 1, y, t) - f(x - 1, y, t)), \tag{B.9}$$

$$f_y(x, y, t) = \frac{1}{2} (f(x, y + 1, t) - f(x, y - 1, t)), \tag{B.10}$$

$$\begin{aligned}
f_{xy}(x, y, t) = \frac{1}{4} & (f(x + 1, y + 1, t) - f(x - 1, y + 1, t) - \\
& f(x + 1, y - 1, t) + f(x - 1, y - 1, t)). \tag{B.11}
\end{aligned}$$

Equations (B.9) and (B.10) are derived using the first order forward and backward difference approximations for $f(x, y, t)$ with respect to x for (B.9) and with respect to y for (B.10), solving each for $f(x, y, t)$, and setting them equal to each other. The process is similar to the way that (A.26) and (A.27) were found in Appendix A. Equation (B.11) is derived using the first order forward and backward difference approximations for $f_x(x, y, t)$ with respect to y , solving each for the like term, $\frac{1}{2} (f(x + 1, y, t) - f(x - 1, y, t))$, and setting them equal to each other. After solving for $f_{xy}(x, y, t)$, the two partial derivative terms on the right side of the equation are substituted using (B.9) as follows:

$$\begin{aligned}
f_{xy}(x, y, t) &= \frac{1}{2}(f_x(x, y + 1, t) - f_x(x, y - 1, t)) \\
&= \frac{1}{2}\left(\frac{1}{2}(f(x + 1, y + 1, t) - f(x - 1, y + 1, t)) - \frac{1}{2}(f(x + 1, y - 1, t) - f(x - 1, y - 1, t))\right). \tag{B.12}
\end{aligned}$$

Simplifying (B.12) gives equation (B.11).

It can be difficult to intuitively appreciate what is truly happening within the context of inpainting when beholding the expanded, discrete representation of the equation. By representing the pixel to be inpainted and its neighboring pixels in a consolidated way, we have a chance at understanding how influential each location is in pixel-value reassignment. Let us now examine the interior of the region to be inpainted. For simplicity, we start by renaming each pixel with a letter as follows:

I	H	G
F	E	D
C	B	A

Figure B-1 Pixel names and locations.

The scheme is detailed in Table B-1.

Table B-1 Letter names associated with pixel locations

A	$f(x + 1, y + 1, t)$	F	$f(x, y - 1, t)$
B	$f(x + 1, y, t)$	G	$f(x - 1, y + 1, t)$
C	$f(x + 1, y - 1, t)$	H	$f(x - 1, y, t)$
D	$f(x, y + 1, t)$	I	$f(x - 1, y - 1, t)$

E	$f(x, y, t)$		
---	--------------	--	--

Proceeding with our naming conventions, we have Table B-2, which presents the associated partial differential equations in terms of the individual pixels in Table B-1.

Table B-2 Partial differential equations in terms of individual pixels

$f_x = \frac{1}{2}(B - H)$	$f_{xy} = \frac{1}{4}(A - G - C + I)$
$f_y = \frac{1}{2}(D - F)$	$f_x^2 = \frac{1}{4}(B^2 - 2BH + H^2)$
$f_{xx} = B - 2E + H$	$f_y^2 = \frac{1}{4}(D^2 - 2DF + F^2)$
$f_{yy} = D - 2E + F$	

Making appropriate substitutions we have:

$$\begin{aligned}
 f_t &= \frac{\frac{1}{4}(B-2E+H)(D-F)^2 + \frac{1}{4}(D-2E+F)(B-H)^2 - \frac{1}{8}(B-H)(D-F)(A-G-C+I)}{\left(\frac{1}{4}(B-H)^2 + \frac{1}{4}(D-F)^2\right)^{\frac{3}{2}}} \\
 &= \frac{2(B+H-2E)(D-F)^2 + 2(D+F-2E)(B-H)^2 - (B-H)(D-F)(A+I-G-C)}{((B-H)^2 + (D-F)^2)^{\frac{3}{2}}}. \tag{B.13}
 \end{aligned}$$

Noting that the pixel values are between 0 and 1 for double precision images, we are looking at a scheme where squaring these pixel values gives small resulting values, also between 0 and 1. The denominator will always be positive and acts as a rescaling factor based on the distances between the vertical pixels, B and H , and horizontal pixels, D and F . The only location the signs of $(B - H)$ and $(D - F)$ matter is in the last term of the numerator.

If there is a diagonal edge forming at the pixel to be inpainted, the values of the neighborhood may resemble the following:

I=.025	H=.025	G=.05
F=.05	E=.05	D=.1
C=.05	B=.1	A=.1

Figure B-2 Sample pixel values and locations corresponding to diagonal edge.

In this case, $B - H$, $D - F$, and $A + I - G - C$ are positive, ensuring the last term of the numerator is positive, and is therefore subtracted from the first two terms. The values of $B + H - 2E$ and $D + F - 2E$ are also positive. The numerator in this example works out to be

$$2(.025)(.05)^2 + 2(.05)(.075)^2 - (.075)(.05)(.025) = .000125 + .0005625 - .00009375$$

$$= .000125 + .0005625 - .00009375$$

$$= .0005975$$

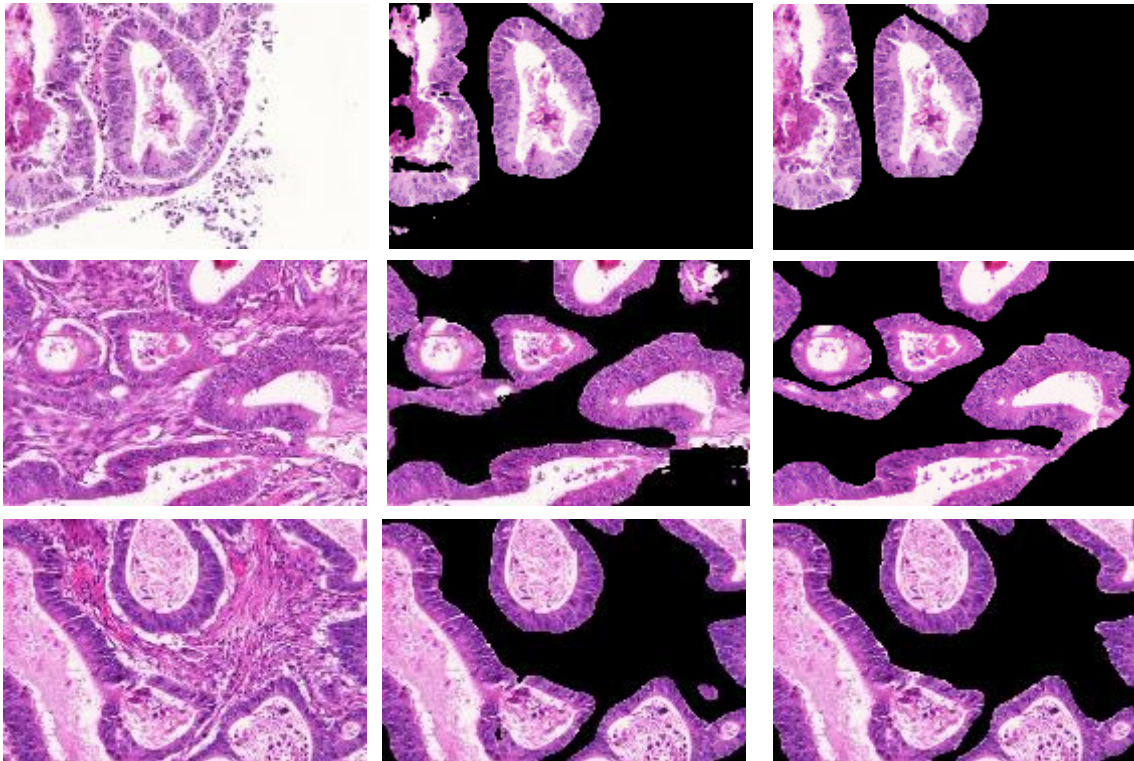
The denominator is $((.05)^2 + (.075)^2)^{\frac{3}{2}} = (.008125)^{\frac{3}{2}} = .0007323776$. Thus, E is assigned the new value .81583598091. With a large value now assigned to E what will happen in subsequent iterations is that E will get exponentially larger (could be in negative or positive direction). To counter this, a term is added to the denominator to make it greater, thus the value of E will be smaller.

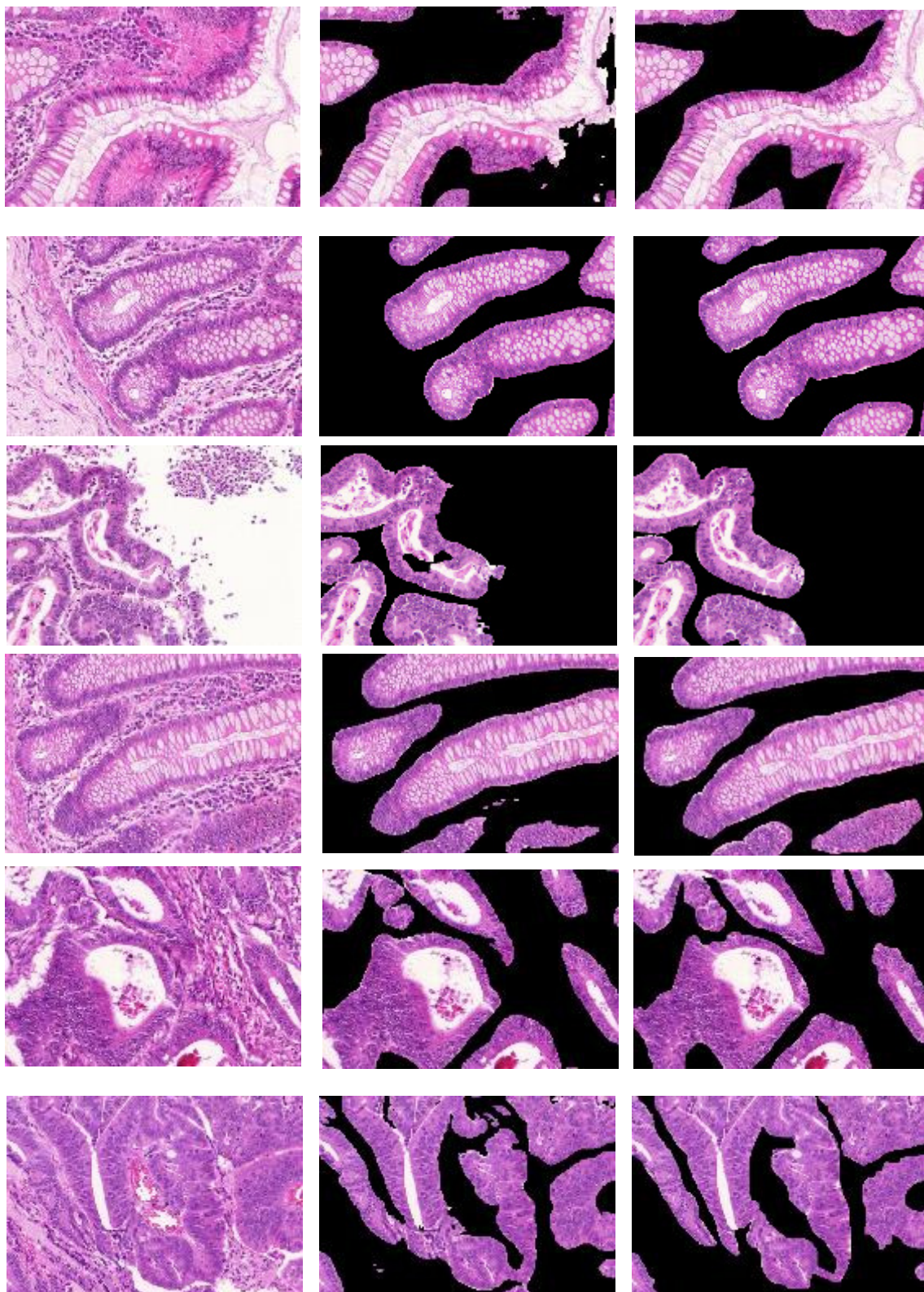
Suppose we add .01 to the denominator in the previous example. The new denominator will be .0107323776, and thus $E = \frac{.0005975}{.0107323776} = 0.0556726592$. Compared to the initial value of $E = .05$, this is a much more suitable reassignment value.

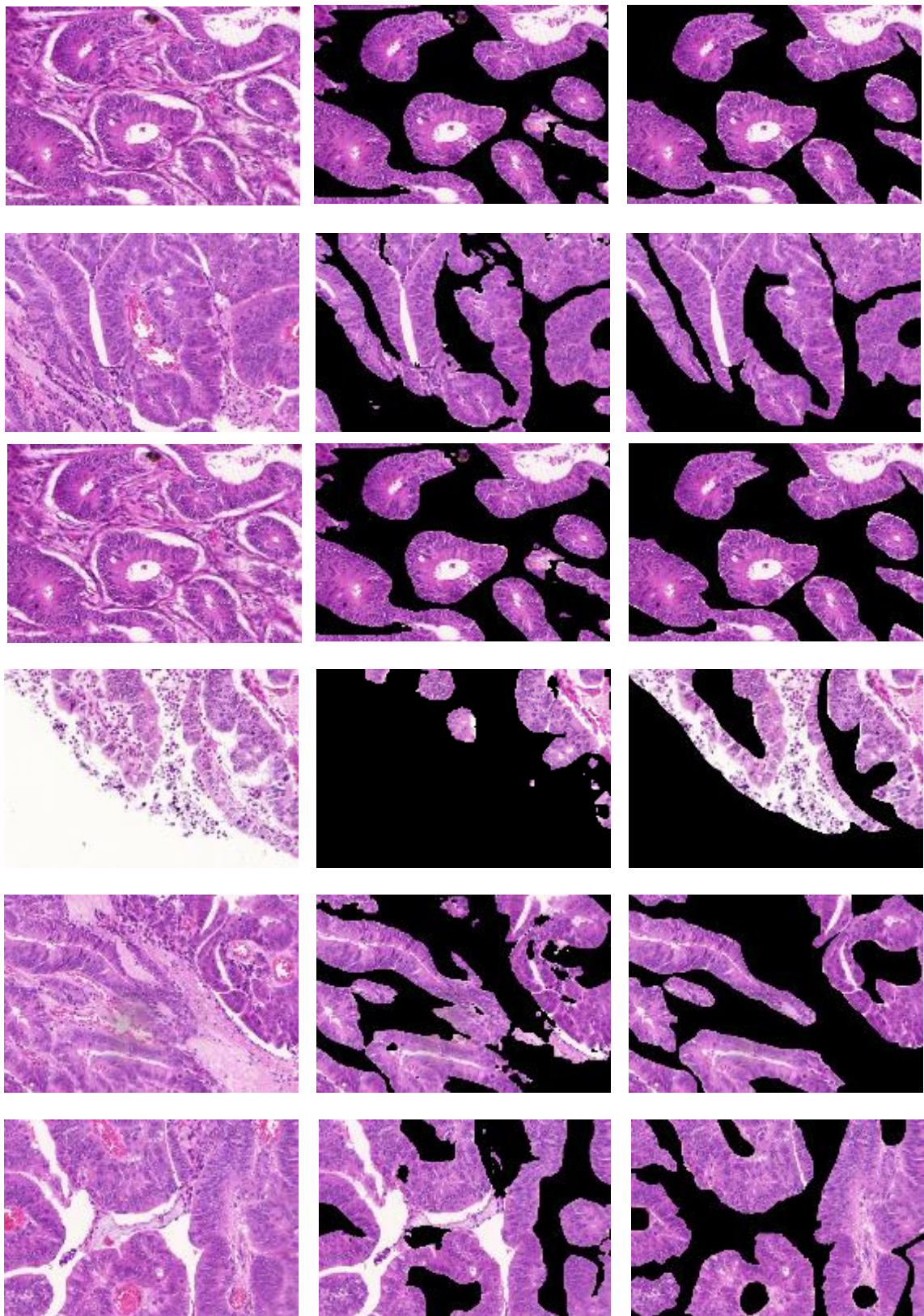
Appendix C. Additional Segmentation Results

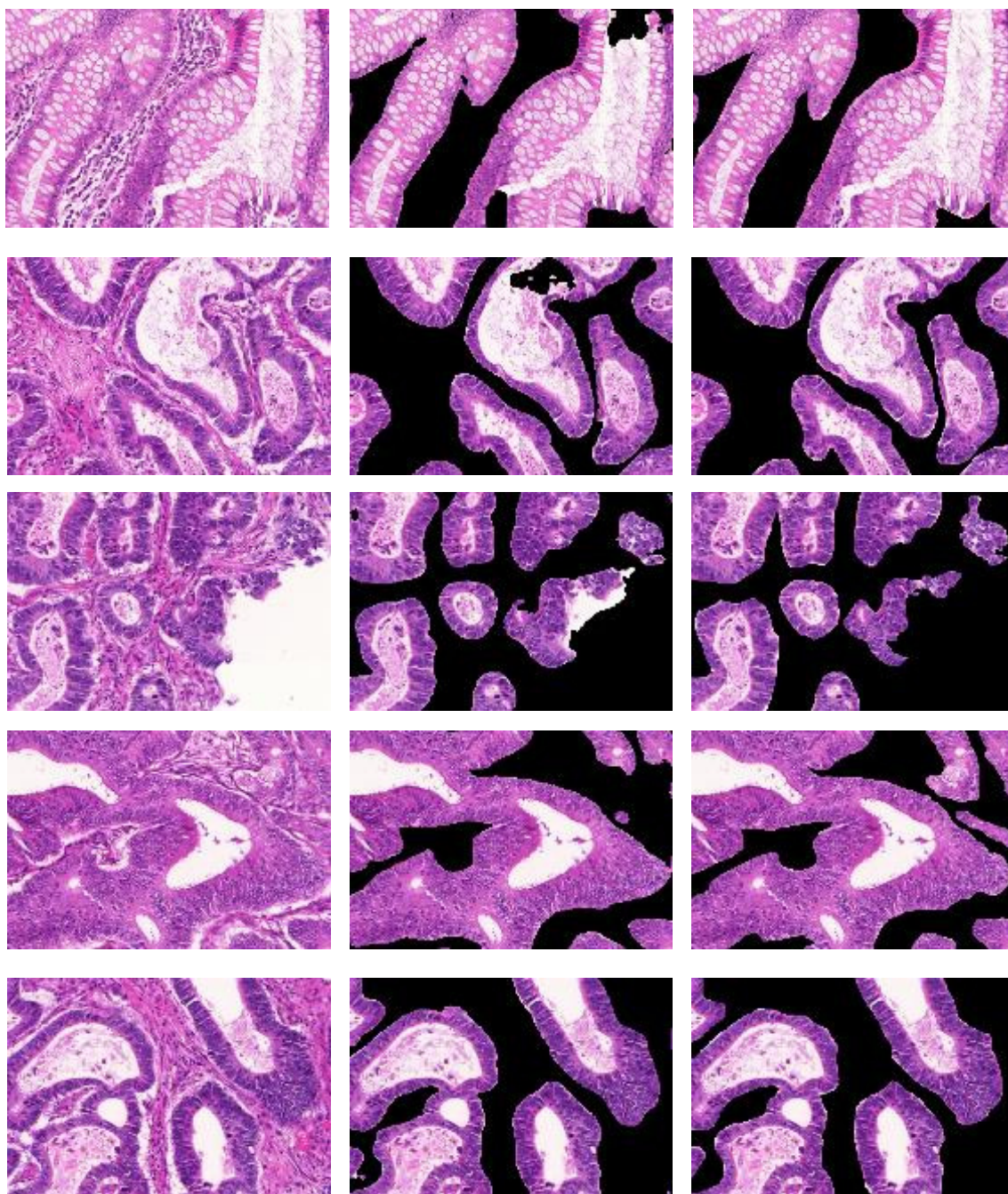
C.1 U-Net Gland Segmentation Results Test Set B

The following images are the gland segmentation results for all twenty images in test set B of [73]. The first column contains input images, the center column contains the U-Net predictions, and the right column contains the ground truth segmentation masks.



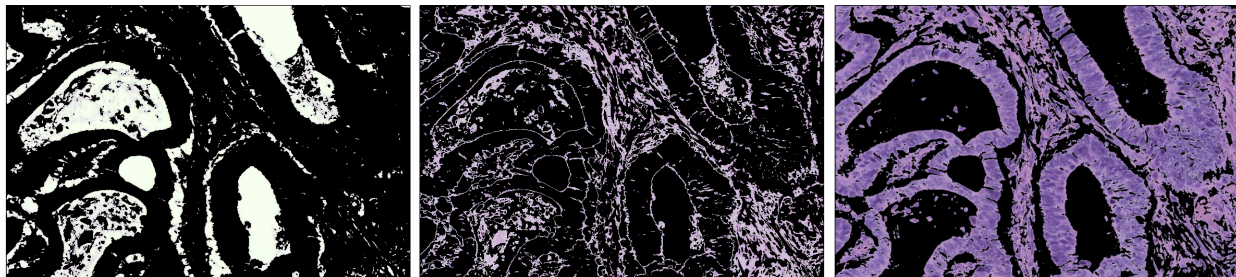






C.2 Histogram Thresholding Results on Image from Test Set B

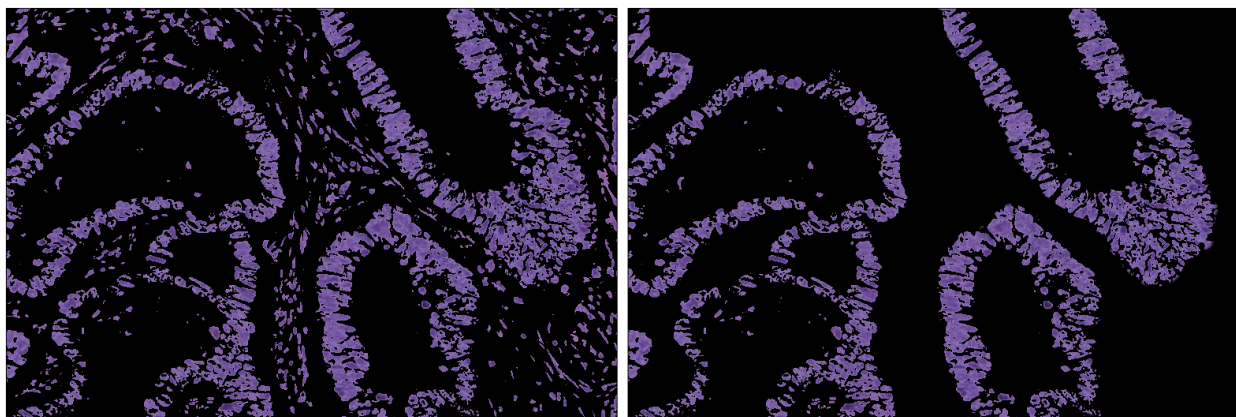
Figure C-1 corresponds to the last set of images in Appendix C.1 after the input image has been adjusted using Reinhard normalization in HistomicsTK [82].



(a) Background artifact isolated using our segmentation method

(b) Stromal tissue isolated using our segmentation method

(c) Tumors and cells in the stroma isolated using our segmentation method



(d) Cell nuclei isolated by transforming the normalized input image to the CIELAB color space, applying the mask in part (c), and removing the light blue pixels from the result

(e) Isolated cell nuclei within the glands by applying the CNN gland segmentation prediction to the results in part (d)

Figure C-1 Segmentation of cell nuclei using our automatic method. From left to right, top to bottom: (a) Slide background and very thin biopsy tissue, (b) stromal tissue with individual cells masked, (c) glands and cells in stroma, (d) isolated cell nuclei within the glands and stroma, and (e) further isolated cell nuclei solely within the glands.

C.3 U-Net and Histogram Thresholding Results on Images from OmniPathology Dataset

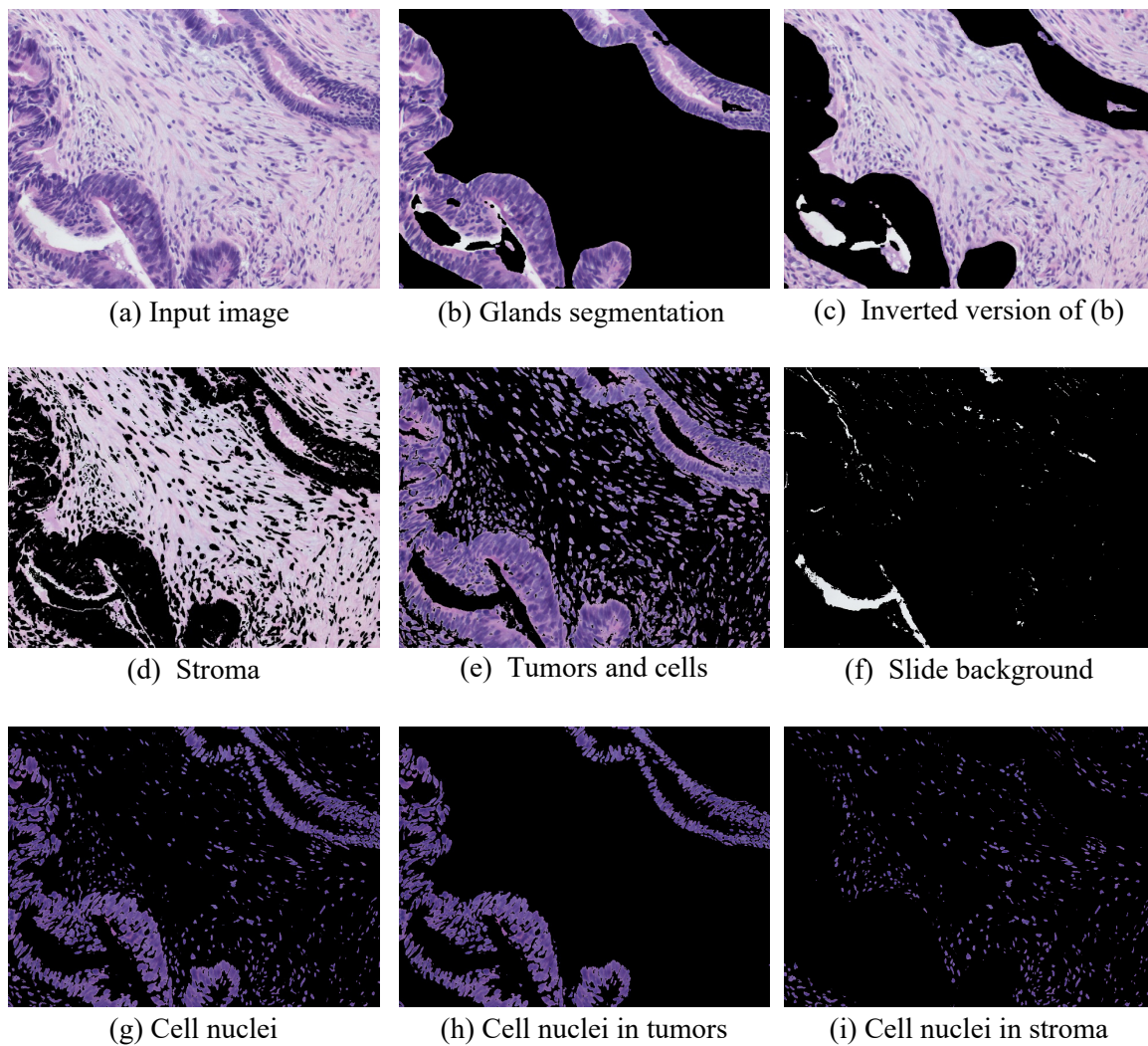


Figure C-2 Two-part segmentation of cell nuclei using our automatic method. From left to right, top to bottom: (a) Region of interest taken from OmniPathology dataset [67] at 20X magnitude, (b) gland segmentation results from trained U-Net, (c) non-glandular structures, (d) stromal tissue with individual cells masked, (e) glands and cells in stroma, (f) slide background, (g) cell nuclei within the glands and stroma, (h) further isolated cell nuclei solely within the glands/tumors, and (i) cell nuclei solely within the stroma. See Figure 6-13 for a close-up comparison of (e) and (i).

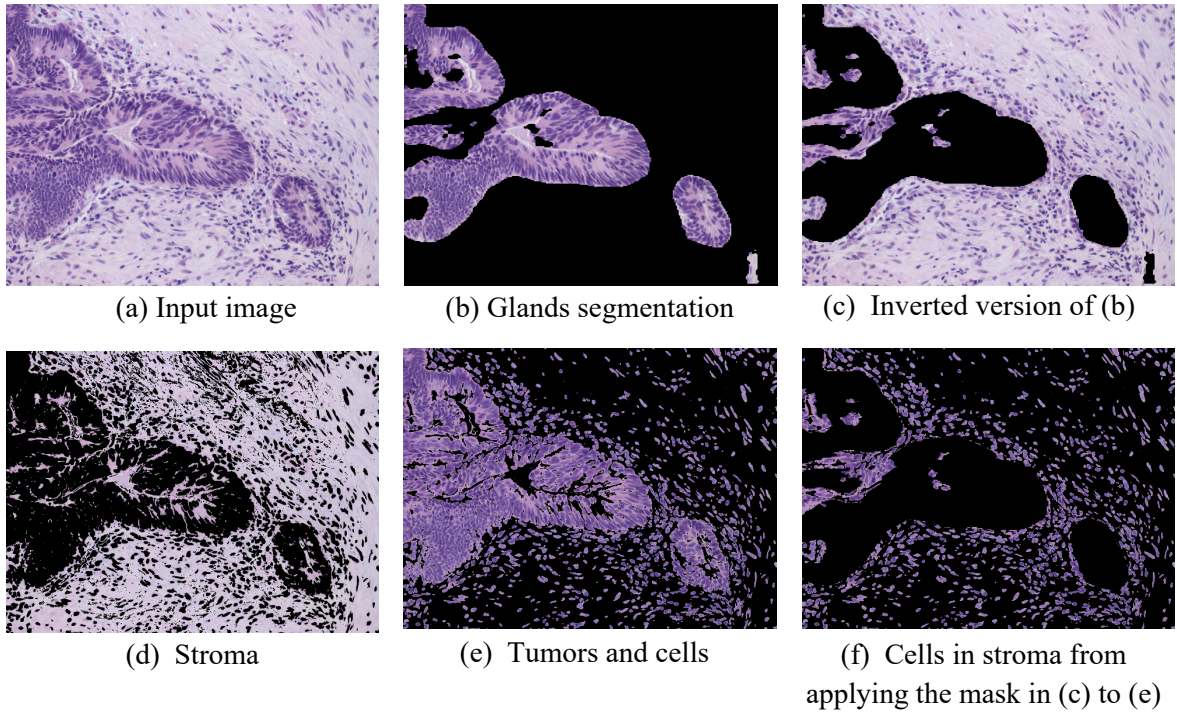


Figure C-3 From left to right, top to bottom: (a) Region of interest taken from OmniPathology dataset [66] at 20X magnitude, (b) gland segmentation results from trained U-Net, (c) non-glandular structures, (d) stromal tissue with individual cells masked, (e) glands and cells in stroma, (f) cells in stroma (tumors masked).

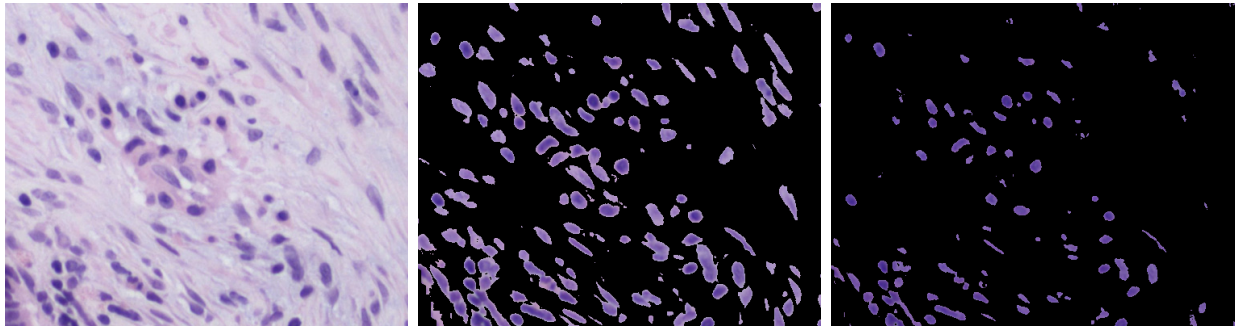


Figure C-4 From left to right: Close-up of input image in Figure C-3, cell segmentation using histogram thresholding algorithm, and cell nuclei segmentation by filtering out light pixels in CIE $L^*a^*b^*$ color space. Both segmentation images are useful depending on the task. Smooth muscle cells are visible in the center image and masked in the second, which favors cells with dark nuclei.

C.4 Automatic thresholding algorithm applied to image of normal colon tissue

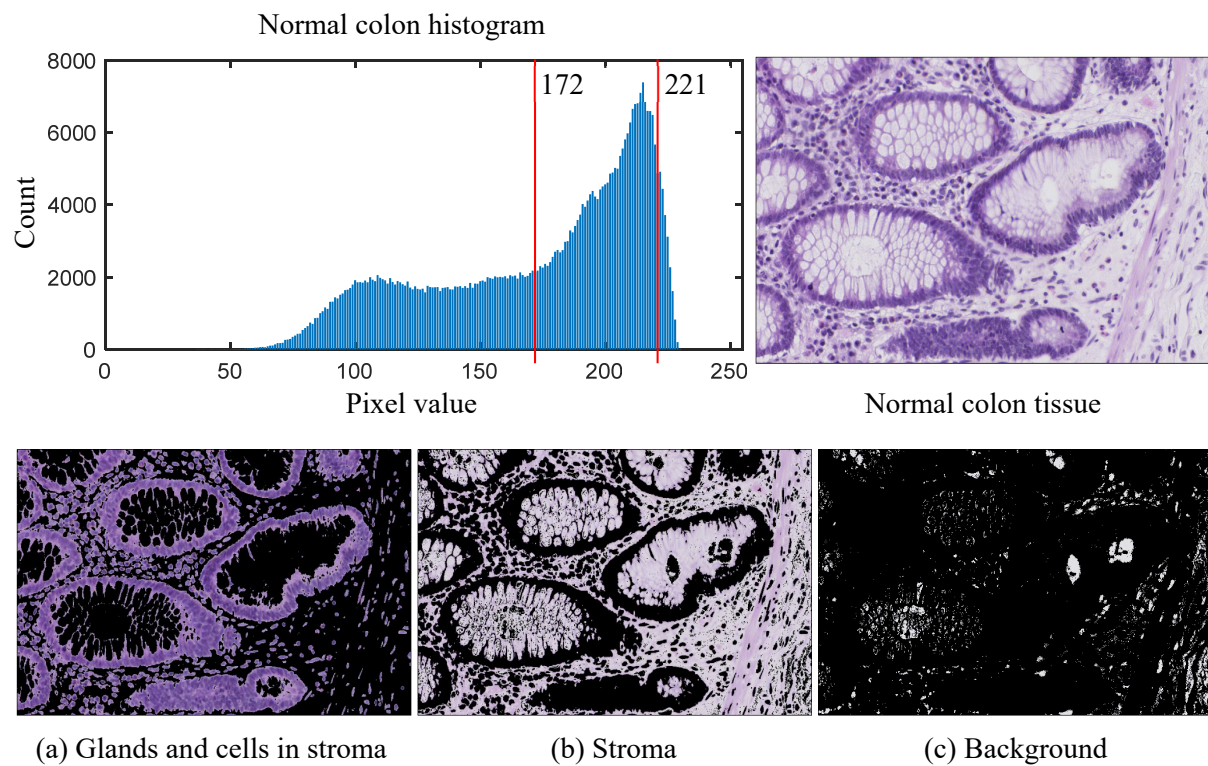


Figure C-5 Histogram and segmentation results for normalized image of benign colon tissue.
Image taken from test set A of [73].

Appendix D. PConvNN Architecture

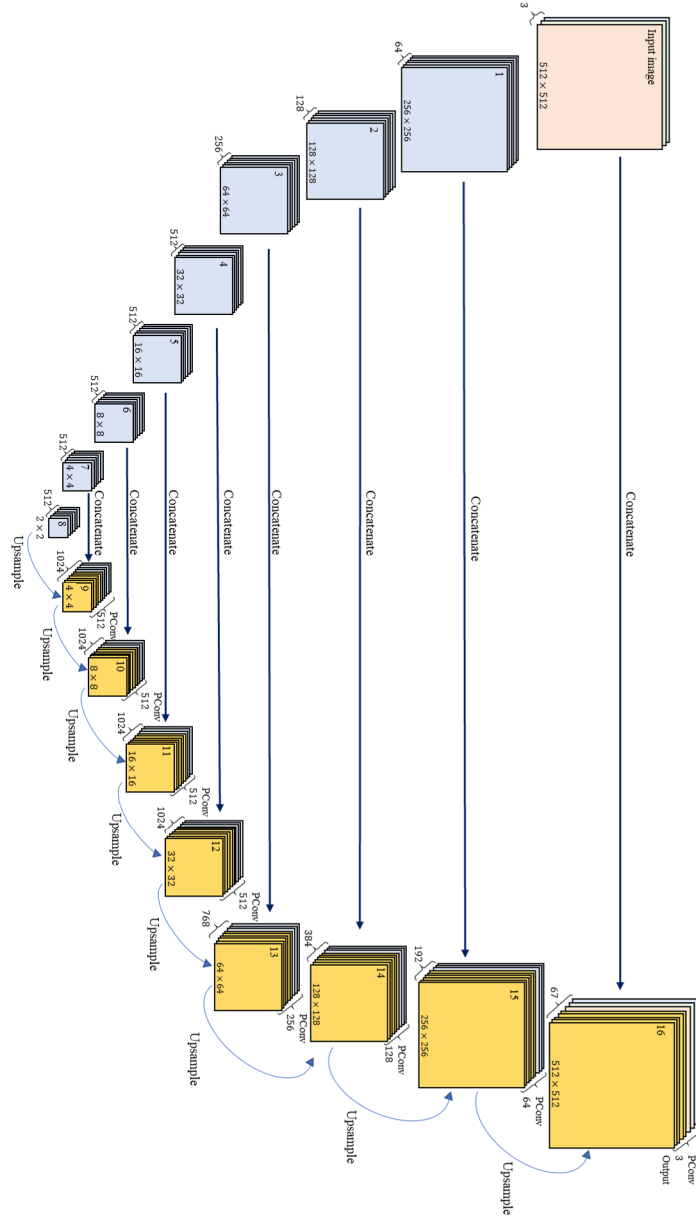


Figure D-1 U-Net-like architecture schematic for region hiding system.